

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure



Release latest for Oracle Cloud
October 2025

ORACLE®

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, Release latest for Oracle Cloud

Copyright © 2020, 2025, Oracle and/or its affiliates.

Primary Author: Anjani Surekha Pothula, Jonah Matishak, Vandana Rajamani

Contributing Authors: German Viscuso, Jeffrey Cowen, Robert Greene, Simon Law, Tejus Subrahmanya

Contributors: German Viscuso, Jeffrey Cowen, Robert Greene, Simon Law, Tejus Subrahmanya

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle®, Java, MySQL, and NetSuite are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1 Overview

New Feature Announcements	1
April 2026	1
March 2026	1
February 2026	3
January 2026	5
About Autonomous AI Database on Dedicated Exadata Infrastructure	6
Key Features of Autonomous AI Database on Dedicated Exadata Infrastructure	7
Deployment Choices for Autonomous AI Database on Dedicated Exadata Infrastructure	8
Components of Autonomous AI Database on Dedicated Exadata Infrastructure	9
User Roles Associated with Autonomous AI Database on Dedicated Exadata Infrastructure	11
Autonomous AI Database Components	11
About Exadata Infrastructure	11
Exadata Infrastructure Requirements	12
Database Features Managed from Exadata Infrastructure	13
Exadata Infrastructure Management Operations	15
About Autonomous Exadata VM Cluster	15
Autonomous Exadata VM Cluster Requirements	16
Database Features Managed from Autonomous Exadata VM Cluster	17
Autonomous Exadata VM Cluster Management Operations	20
About Autonomous Container Database	21
Autonomous Container Database Requirements	22
Database Features Managed from Autonomous Container Database	22
Autonomous Container Database Management Operations	29
Monitoring Autonomous Container Database	29
About Autonomous AI Database	30
Autonomous AI Database Requirements	30
Database Features Managed from Autonomous AI Database	30
Autonomous AI Database Tools	35
Autonomous AI Database Management Operations	37
Autonomous AI Database on Dedicated Exadata Infrastructure in AWS Region	38
Prerequisites	39
Creating Autonomous AI Database Resources in AWS Region	39

Managing Oracle Database@AWS Resources	40
Observe Oracle Database@AWS	42
Unsupported Features in Oracle Database@AWS	42
Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@Azure	42
Prerequisites	42
Creating Autonomous AI Database Resources in Azure Region	43
Managing Oracle Database@Azure Resources	43
Unsupported Features in Oracle Database@Azure	45
Autonomous AI Database for Developers	45
Requirements	45
Supported Oracle Database Versions	45
Provisioning Workflow	45
Specifications	46
Excluded Features	46
Supported Features	46
Billing Autonomous AI Database on Dedicated Exadata Infrastructure	47
CPU Billing Details	47
Elastic Pool Billing	48

2 Quick Start

Before You Begin with Autonomous AI Database	1
Request Service Limit Increase	2
To Request a Service Limit Increase	3
Access Autonomous AI Database in the Oracle Cloud Infrastructure Console	4
Obtain Tenancy Details	5
Database Name	6
Region	6
Tenancy OCID	6
Database OCID	6
Compartment OCID	6
Outbound IP Address	6
Availability Domain	7
Get Help and Contact Oracle Support	7
Create a Service Request in My Oracle Cloud Support	7
Register to My Oracle Cloud Support as a New User	9
Create a Support Ticket from Oracle Cloud Infrastructure console	10
LiveLabs for Autonomous AI Database on Dedicated Exadata Infrastructure	10
Workflows	11
Set up Infrastructure Resources for Autonomous AI Database on Dedicated Exadata Infrastructure	11
Create and Use Autonomous AI Database	12

3 Features

High Availability	1
Protect Critical Databases from Failures and Disasters Using Autonomous Data Guard	1
About Autonomous Data Guard	1
Role Transitions and Operations	3
Automatic Failover or Fast-Start Failover	4
Snapshot Standby Database	5
Monitoring Lag Times	7
Autonomous Data Guard Configuration Options	7
Best Practices while Configuring Autonomous Data Guard	9
How Autonomous Data Guard Affects Standard Management Operations	9
Step-by-Step Guides	10
Backup and Restore Autonomous AI Database on Dedicated Exadata Infrastructure	11
About Backup and Recovery	11
Backup and Restore Operations	19
Backup and Restore Events	21
Best Practices for estimating Backup space	21
Use OCI Full Stack Disaster Recovery on Autonomous AI Database on Dedicated Exadata Infrastructure	23
Prerequisite Steps for Autonomous Container Database with Full Stack Disaster Recovery	23
Enable Full Stack Disaster Recovery	23
Use Full Stack Disaster Recovery to Perform Switchover or Failover Operations	24
High Performance	24
High Performance Features in Autonomous AI Database on Dedicated Exadata Infrastructure	24
Predefined Database Services	25
Connection Pools	26
Special-Purpose Connection Features	26
SQL Performance Tuning Features	27
SQL Tracing	27
Optimizer Statistics	27
Optimizer Hints	28
Automatic Indexing	29
Fast Ingest	30
Predefined Job Classes with Oracle Scheduler	30
Performance Monitoring and Tuning Tools	31
Use SQL Tracing on Autonomous AI Database on Dedicated Exadata Infrastructure	32
Configure SQL Tracing on Autonomous AI Database	33
Enable SQL Tracing on Autonomous AI Database	34

Disable SQL Tracing	35
View Trace File Saved to Cloud Object Store on Autonomous AI Database	35
View Trace Data in SESSION_CLOUD_TRACE View on Autonomous AI Database	36
Governance	37
Tagging Autonomous AI Database Cloud Resources	37
Importance of Tagging	37
Adding Tags	38
Oracle Standard Tags	39
Oracle Application Name Tags	42
Manage Resource Availability with Compartment Quotas	42
Quotas for Autonomous AI Database on Oracle Public Cloud	43
Quotas for Autonomous AI Database on Exadata Cloud@Customer	44
Continuous Availability	45
Code for Continuous Availability	45
Application Configuration Checklist	46
Connect Using Database Services	46
Configure Connection String for High Availability	46
Use Fast Application Notification (FAN)	47
Use Recommended Practices to Allow Draining	48
Enable Transparent Application Continuity (TAC) or Application Continuity (AC)	49
Configure Drivers for Continuous Availability	51
Configure JDBC Thin Driver	51
Configure OCI (Oracle Call Interface) Driver	52
Configure ODP.NET Unmanaged Provider Driver	53
Enable Service Attributes for Failover	53
Configure Service Attributes for Maintenance	54
Configure Clients for FAN Including Optional Wallets	54
Compute Management	56
Compute Models in Autonomous AI Database on Dedicated Exadata Infrastructure	56
Compute Management	57
CPU Allocation When Auto-Scaling	58
How Compartment Quotas Affect CPU Management	59
How VM Cluster Nodes Affect CPU Management	60
Consolidate Autonomous AI Database Instances Using Elastic Pools	62
About Elastic Pools	62
Benefits of Using Elastic Pools	63
Requirements to Create an Elastic Pool	63
Requirements to Join an Elastic Pool	64
Pool Leader and Member Instance ECPU Allocation	64
Pool Capacity for an Elastic Pool	64
Elastic Pool Leader Operations	65
Elastic Pool Member Operations	65

Change Management	66
Service Maintenance of Autonomous AI Database on Dedicated Exadata Infrastructure	66
Service Maintenance Types	66
Specify When Maintenance Can Occur	68
Specify What Kind of Patches Apply	73
View and Manage Already Scheduled Maintenance	73
View Maintenance Status Notifications	75
Automatic Queuing of Maintenance Events	76
View Past Maintenance Events	77
Monitor Service Maintenance Events	77
Observability	78
Plan and Observe Capacity	78
Resource Terminology	78
Resource Limits	79
Resource Usage Tracking	80
Resource Usage Visualizations	81
Exadata System Shapes	84
Database Observability with Autonomous AI Database Metrics	85
Prerequisites	86
Using the Console	86
Using the API	86
Available Metrics: oci_autonomous_database	87
Database Observability with Performance Hub	93
Database Observability with Dynamic Performance Views	95
Dynamic Performance Views	95
Accessing Dynamic Performance Views at the Autonomous Container Database Level	96
Supported Dynamic Performance Views at the Autonomous Container Database Level	96
Example Queries	97
Use Database Actions to Manage and Observe Databases	98
Use Database Management Service to Observe Databases	99
Use Enterprise Manager to Manage and Observe Databases	100
Use Ops Insights on Autonomous AI Database	101
About Ops Insights	101
Use Work Requests to Observe Long-Running Operations	102
Use Events and Notifications to Watch Resource Activity	102
Observe Autonomous AI Database on Dedicated Exadata Infrastructure Resource Logs with OCI Logging Service	103
Security	108
Security Features in Autonomous AI Database on Dedicated Exadata Infrastructure	108
Configuration Management	109
Data Encryption	109

Auditing	110
Access Control Within Autonomous AI Database on Dedicated Exadata Infrastructure	110
Certificate Management	110
Data Protection	113
Sensitive Data Discovery and Data Masking	114
Regulatory Compliance Certification	115
Configuration Management in Autonomous AI Database on Dedicated Exadata Infrastructure	115
Autonomous Virtual Machine Hardening	116
Configuration Drift Management	116
File Integrity and Intrusion Monitoring	117
Autonomous VM Vulnerability Scanning and Response	117
Vulnerability Scans and Patching	118
Oracle Cloud Security Testing Policy	119
Endpoint, Malware, and Ransomware Protection	119
Security Incident Management	119
Data Encryption in Autonomous AI Database on Dedicated Exadata Infrastructure	120
Encryption of Data at Rest	120
Encryption of Data in Transit	120
Auditing Capabilities in Autonomous AI Database	121
Auditing Service Level Activities	121
Auditing Database Activities	122
Auditing Autonomous VM Activities	123
Auditing Oracle Operator Activities	124
Access Control Within Autonomous AI Database on Dedicated Exadata Infrastructure	125
Oracle Cloud User Access Control	125
Client Access Control	127
Database User Access Control	130
Privileged Access Management (PAM)	132
IAM Policies for Autonomous AI Database on Dedicated Exadata Infrastructure	133
Policy Details for Autonomous AI Database	133
Policies to Manage Exadata Infrastructure Resources	139
Policies to Manage Autonomous Exadata VM Clusters	139
Policies to Manage Autonomous Container Databases	140
Policies to Manage Autonomous Data Guard Configuration	141
Policies to Manage Autonomous AI Databases	141
Master Encryption Keys in Autonomous AI Database on Dedicated Exadata Infrastructure	142
Use Customer-Managed Keys in the Vault Service	143
Use Bring Your Own Keys (BYOK) in Vault Service	144
Use External Keys from OCI External Key Management Service (OCI EKMS)	145
Use Customer-Managed Keys in AWS KMS	146

Create a Service Gateway, a Route Rule and an Egress Security Rule	146
Create a Dynamic Group and a Policy Statement	147
Use Customer-Managed Keys in Oracle Key Vault	148
Features for Experienced Database Users	149
Oracle Database Features in Autonomous AI Database on Dedicated Exadata Infrastructure	149
Latest Autonomous AI Database Features	150
The ADMIN User and the SYS User	151
Database Initialization Parameters	152
SYSDATE_AT_DBTIMEZONE Select a Time Zone for SYSDATE on Autonomous AI Database	154
Oracle Database Features with Limited Support in Autonomous AI Database on Dedicated Exadata Infrastructure	156
SQL Commands	157
Data Types	160
Database In-Memory	160
Manually Created External Tables	161
Oracle LogMiner	161
Oracle Text	162
Oracle XML DB	163
Oracle Flashback and Restore Points	164
Database Replay	164
Database PL/SQL Packages	165
Oracle Database Features That Are Not Supported in Autonomous AI Database on Dedicated Exadata Infrastructure	166
Character Set Selection for Autonomous AI Database	167
Use Oracle Graph with Autonomous AI Database on Dedicated Exadata Infrastructure	168
Build Applications Using Graph APIs	168
Use Oracle Spatial in Autonomous AI Database	169
About Oracle Spatial	169
Oracle Cloud Marketplace Images for Spatial	169
Use Advanced Queuing with Autonomous AI Database	170
Use JSON Documents with Autonomous AI Database on Dedicated Exadata Infrastructure	170
Use MongoDB API with Autonomous AI Database on Dedicated Exadata Infrastructure	172
Enable MongoDB API	173
Connect MongoDB Applications	173
Test MongoDB Connections	174

4 Tasks

About Connecting to Autonomous AI Database on Dedicated Exadata Infrastructure	1
Oracle Call Interface (OCI) Connections and Wallets	2

Predefined Database Service Names for Autonomous AI Databases	2
Connect to Autonomous AI Database Using a Client Application	5
Connect to Autonomous AI Database Using Oracle Database Tools	6
Connect with Microsoft .NET, Visual Studio, and Visual Studio Code	6
Connect to Autonomous AI Database with Scripting Languages	7
Connect with Oracle Cloud Infrastructure FastConnect	7
Migrate	8
Attach External Storage to Autonomous AI Database	8
Attach Network File Storage to Autonomous AI Database on Dedicated Exadata Infrastructure	8
Attach External File Storage to Autonomous AI Database on Dedicated Exadata Infrastructure	12
Move Data Into Autonomous AI Database	17
Migrate Data into Autonomous AI Database on Dedicated Exadata Infrastructure	17
Loading Data into Autonomous AI Database on Dedicated Exadata Infrastructure	22
Load Data from Files in the Cloud	26
Use Oracle Data Pump	36
Use SQL*Loader	41
Migrate to Autonomous AI Database Using Oracle Database Migration Service	42
Use Oracle GoldenGate to Replicate Data with Autonomous AI Database	42
Decrypt Data While Importing from Object Storage	44
Export Data from Autonomous AI Database	48
Export Data from Autonomous AI Database on Dedicated Exadata Infrastructure	48
Use Oracle Data Pump to Export Data from Autonomous AI Database	48
Export Data to a Directory	55
Export Data to Object Store as Text Using DBMS_CLOUD.EXPORT_DATA	62
Encrypt Data While Exporting to Object Storage	72
Use Data Pipelines for Continuous Load and Export	76
About Data Pipelines on Autonomous AI Database on Dedicated Exadata Infrastructure	76
Create and Configure Pipelines	80
Control Pipelines (Start, Stop, Drop, or Reset a Pipeline)	86
Bulk Operations for Files in the Cloud	93
About Bulk File Operations	93
Bulk Copy Files in Cloud Object Storage	94
Bulk Move Files Across Cloud Object Storage	95
Bulk Download Files from Cloud Object Storage	96
Bulk Upload Files to Cloud Object Storage	96
Bulk Delete Files from Cloud Object Storage	97
Monitor and Troubleshoot Bulk File Loads	98
Manage Users	100
Managing Database Users	100
Create Database Users	100

Unlock or Change the ADMIN Database User Password	101
Manage Database User Privileges	101
Remove Database Users	102
Manage User Profiles with Autonomous AI Database	103
Manage Password Complexity on Autonomous AI Database	104
Gradual Database Password Rollover for Applications	105
Use Oracle Database Vault to Manage Database User Privileges	106
Configure and Enable Oracle Database Vault	106
Use Identity and Access Management (IAM) Authentication with Autonomous AI Database	107
About Identity and Access Management (IAM) Authentication with Autonomous AI Database	107
Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous AI Database	108
Change External Identity Providers on Autonomous AI Database	110
Create Identity and Access Management (IAM) Groups and Policies for IAM Users	111
Add IAM Users on Autonomous AI Database	113
Add IAM Roles on Autonomous AI Database	114
Create IAM Database Password for IAM Users	115
Connect to Autonomous AI Database with Identity and Access Management (IAM) Authentication	115
Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password	117
Configuring a Client Connection for SQL*Plus That Uses an IAM Token	118
Use Instance Principal to Access Autonomous AI Database with Identity and Access Management (IAM) Authentication	120
Configure Proxy Authentication	120
Use Database Link with IAM Authenticated Users	122
Disable Identity and Access Management (IAM) Authentication on Autonomous AI Database	122
Notes for Using Autonomous AI Database Tools with Identity and Access Management (IAM) Authentication	122
About Kerberos Authentication	123
Components of the Kerberos Authentication System	123
Notes About Kerberos Authentication on Autonomous AI Database	124
Enable Kerberos Authentication on Autonomous AI Database	125
Disable Kerberos Authentication on Autonomous AI Database	127
Authenticate and Authorize Microsoft Azure AD Users for Autonomous AI Database	127
About Integrating Oracle Autonomous AI Database on Dedicated Exadata Infrastructure with Microsoft Azure AD	127
General Process of Authenticating Microsoft Azure AD Identities with Oracle Autonomous AI Database on Dedicated Exadata Infrastructure	129
Enable Azure AD Authentication on Autonomous AI Database	130
Map Oracle Database Schemas and Roles	134

Configure Client Connections to Azure ADs	134
Troubleshooting Microsoft Entra ID Connections	135
Use Microsoft Active Directory with Autonomous AI Database on Dedicated Exadata Infrastructure	136
Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous AI Database	136
Configure CMU with Microsoft Active Directory on Autonomous AI Database	137
Configure CMU with Microsoft Active Directory on Exadata Cloud@Customer	140
Add Microsoft Active Directory Roles on Autonomous AI Database	143
Add Microsoft Active Directory Users on Autonomous AI Database	144
Connect to Autonomous AI Database with Active Directory User Credentials	146
Verify Active Directory User Connection Information with Autonomous AI Database	146
Remove Active Directory Users and Roles on Autonomous AI Database	147
Disable Active Directory Access on Autonomous AI Database	148
Limitations with Microsoft Active Directory on Autonomous AI Database	148
Develop	148
Developing Applications with Autonomous AI Database	148
Development Workflow	149
Create a Database	149
Create a Directory	149
Manage Directories	150
Configure a Development System to Access the Database	152
Define or Load the Database Structure	153
Download, Install and Configure Oracle Database JDBC Driver	154
Download, Install and Configure Oracle Instant Client	154
The Basics of Building an Application	155
Build Database Applications with Autonomous AI Database	156
Build a Java Application	156
Build a .NET Application with Visual Studio and Visual Studio Code	159
Build a Node.js Application	161
Build a PHP Application	164
Build a Python Application	167
Build a Ruby Application	170
Build an APEX (Oracle Application Express) Application	172
Developing RESTful Services in Autonomous AI Database	176
Send Email and Notifications from Autonomous AI Database	177
Send Email on Autonomous AI Database on Dedicated Exadata Infrastructure	177
Send Microsoft Teams Notifications from Autonomous AI Database on Dedicated Exadata Infrastructure	184
Send Slack Notifications from Autonomous AI Database on Dedicated Exadata Infrastructure	188
Make External Calls using a Customer-Managed Wallet	191
Prepare the Customer-Managed Wallet	192

Use a customer-managed wallet with UTL_HTTP	192
Use a Customer-Managed Wallet for Scheduler Email Notifications (SMTP)	195
Manage and Store Files in a Cloud Code Repository with Autonomous AI Database	197
About Cloud Code Repositories with Autonomous AI Database	198
Initialize a Cloud Code Repository	198
Create and Manage a Cloud Code Repository	200
Create and Manage Branches in a Cloud Code Repository	201
Export Schema Objects to the Cloud Code Repository Branch	203
Use File Operations with a Cloud Code Repository	204
Use SQL Install Operations with a Cloud Code Repository	205
Cache Messages with Singleton Pipes	206
About Caching Messages with Singleton Pipes	206
Automatic Refresh of Cached Message with a Cache Function	209
Create an Explicit Singleton Pipe	210
Create an Explicit Singleton Pipe with a Cache Function	212
Use Persistent Messaging with Messages Stored in Cloud Object Store	213
About Persistent Messaging with DBMS_PIPE	213
Create an Explicit Persistent Pipe and Send a Message	215
Retrieve a Persistent Message on Same Database	217
Retrieve a Persistent Message by Creating a Pipe on a Different Database	219
Remove a Persistent Pipe	221
Invoke Cloud Functions and External Procedures	221
About Cloud Functions and External Procedures	222
Invoke OCI Cloud Functions as SQL Functions	222
Invoke AWS Lambda Functions as SQL Functions	228
External Procedures Overview	229
System Configuration for External Procedures and Scripts	229
Implement User Defined Functions with External Procedures	230
Manage Data	238
About Cloning Autonomous Container Database on Dedicated Exadata Infrastructure	238
About Cloning Autonomous AI Database on Dedicated Exadata Infrastructure	240
Clone Types	240
Clone Sources	240
Clone Requirements	241
Cross Tenancy Clone Requirements	242
Clone Limitations	244
Step-by-Step Guides	245
Create and Manage External Table Cache in Autonomous AI Database	245
About External Table Cache in Autonomous AI Database	245
Prerequisites	246
Limitations	246
Create External Table Cache	246

Populate Files into External Table Cache	248
Drop Files from External Table Cache	250
Disable and Enable External Table Cache	252
Drop External Table Cache	253
View External Table Cache Information	254
Set Optional Sizing Preferences for External Table Cache	254
External Table Metadata Columns	255
Use Oracle Workspace Manager in Autonomous AI Database	256
Link Data	256
Create TLS Database Links from an Autonomous AI Database to another Autonomous AI Database	256
Create Non-TLS Database Links from Autonomous AI Database to an Autonomous AI Database	262
Create Database Links from an Autonomous AI Database to a Publicly Accessible Oracle Database with a Wallet (mTLS)	265
Create Database Links from Autonomous AI Database to an Oracle Database on a Private Endpoint	269
Create Database Links with Customer-Managed Heterogeneous Connectivity to Publicly Accessible Non-Oracle Databases	277
Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint	281
Querying External Data with Autonomous AI Database on Dedicated Exadata Infrastructure	286
Query External Data on Oracle Compute Cloud at Customer (C3)	291
Query External Data with Data Catalog	293
Query External Data with AWS Glue Data Catalog	307
About Querying Apache Iceberg Tables	314
Integrate	318
Analyzing and Visualizing Data	318
Use Oracle Analytics Desktop to Analyze and Visualize Data	318
Use Oracle Analytics Cloud to Analyze and Visualize Data	319
Oracle Extensions for IDEs	320
Use Oracle Cloud Infrastructure Toolkit for Eclipse	320
Use Oracle Developer Tools for Visual Studio	320
Use Oracle Developer Tools for VS Code	321
Connect Power BI and Microsoft Data Tools to Autonomous AI Database on Dedicated Exadata Infrastructure	322
Use Select AI for Natural Language Interaction with your Database	323
Use Select AI to Generate SQL from Natural Language Prompts	323
Select AI with Retrieval Augmented Generation (RAG)	390
Synthetic Data Generation	403
Select AI Agent	413

5 Use Cases

Configuration Use Cases	1
Configure Autonomous AI Database for Proof of Concept (POC)	1
Use Case	1
Resources Needed	2
High-Level Steps	4
Step 1. Create OCI IAM Components	5
Step 2. Create the VCN and Subnets	7
Step 3. Assign Fleet Administrator	8
Step 4. Create Autonomous AI Database Resources	9
Configure Autonomous AI Database with Reference Architecture	10
Use Case	10
Resources Needed	11
High-Level Steps	14
Step 1. Create Compartments	15
Step 2. Create Groups	15
Step 3. Create Policies	15
Step 4. Create the VCN and Subnets	17
Step 5. Assign Fleet Administrator	20
Step 6. Create the Exadata Infrastructure Resource	20
Step 7. Create the Autonomous Exadata VM Cluster Resources	20
Step 8. Assign Application DBAs	20
Step 9. Create Autonomous Container Database Resources	21
Step 10. Create Autonomous AI Database Resources	21
Notifications Example: Emails for Maintenance Events	21
Step 1. Create a Notifications Topic	22
Step 2. Add a Subscription to the Notifications Topic	23
Step 3. Add an Events Service Rule	23

6 How-to Guides

Create and Manage Exadata Infrastructure Resources	1
Create an Exadata Infrastructure Resource	1
Manage Exadata Infrastructure	4
View a List of Exadata Infrastructure Resources	4
View Details of an Exadata Infrastructure Resource	5
Move an Exadata Infrastructure Resource to a Different Compartment	7
Manage Customer Contacts for an Exadata Infrastructure	8
Terminate an Exadata Infrastructure Resource	9
Manage Exadata Infrastructure Maintenance	9
View and Manage Scheduled Maintenance of an Exadata Infrastructure Resource	9

Change the Maintenance Schedule of an Exadata Infrastructure Resource	10
View Past Maintenance of an Exadata Infrastructure Resource	11
Create and Manage Autonomous Exadata VM Clusters	11
Create Autonomous Exadata VM Cluster	11
IAM Policy Requirements	11
Zero Trust Packet Routing (ZPR) Policy Requirements	11
Resource Requirements	12
Network Requirements	13
Instructions	16
Formula for sizing AVMC resources	26
Manage Autonomous Exadata VM Clusters	27
View a List of Autonomous Exadata VM Clusters	28
View Details of an Autonomous Exadata VM Cluster	30
Change the License Type on an Autonomous VM Cluster	34
Move an Autonomous Exadata VM Cluster to Another Compartment	34
Enable or Disable AWS KMS	35
Manage Security Certificates for an Autonomous Exadata VM Cluster Resource	36
View Resource Usage for an Autonomous Exadata VM Cluster	38
Manage Autonomous Exadata VM Cluster Resources	40
Terminate an Autonomous Exadata VM Cluster	40
Manage Exadata Infrastructure Maintenance	41
Change the Maintenance Schedule of an Autonomous Exadata VM Cluster	41
View and Manage Scheduled Maintenance of an Autonomous Exadata VM Cluster	42
View Past Maintenance of an Autonomous Exadata VM Cluster	43
Create and Manage Autonomous Container Databases	43
Create an Autonomous Container Database	43
Edit Autonomous Container Database Backup Settings	51
Configure and Manage Autonomous Data Guard	55
Enable Autonomous Data Guard on an Autonomous Container Database	55
View the Status of an Autonomous Data Guard Configuration	57
Add a Second Standby Autonomous Container Database	58
Switch Roles in an Autonomous Data Guard Configuration	59
Fail Over to the Standby in an Autonomous Data Guard Configuration	60
Reinstate the Disabled Standby in an Autonomous Data Guard Configuration	60
Update Autonomous Data Guard Settings	61
Convert Physical Standby to Snapshot Standby	62
Convert Snapshot Standby to Physical Standby	63
Add a Cross Tenancy Standby Database	63
Manage Customer Contacts for an Autonomous Container Database	69
Add Customer Contacts	70
Edit Customer Contacts	70
Delete Customer Contacts	70

Move an Autonomous Container Database to a Different Compartment	70
Restart an Autonomous Container Database	71
Schedule a Quarterly Maintenance Update	71
Terminate an Autonomous Container Database	73
Update Autonomous Container Database Maintenance Preferences	74
View and Manage Scheduled Maintenance of an Autonomous Container Database	75
View Details of an Autonomous Container Database	76
View a List of Autonomous Container Databases	78
View a List of Autonomous Container Databases in a Compartment	78
View a List of Autonomous Container Databases in an Autonomous Exadata VM Cluster Resource	80
View NFS Space Utilization	82
View Past Maintenance of an Autonomous Container Database	82
View Resource Usage for an Autonomous Container Database	83
Clone an Autonomous Container Database	84
Clone an Autonomous Container Database	84
Clone an Autonomous Container Database from a Remote Backup	89
Create and Manage Autonomous AI Databases	95
Create an Autonomous AI Database	95
Backup an Autonomous AI Database on Dedicated Exadata Infrastructure Manually	102
Clone an Autonomous AI Database on Dedicated Exadata Infrastructure	102
Clone an Autonomous AI Database Instance	103
Clone Autonomous AI Database from Backup	107
Cross Tenancy Clone an Autonomous AI Database Instance	114
Cross Tenancy Clone an Autonomous AI Database from Backup	116
Create a Long-Term Backup	118
Procedure	119
Enable or Disable Auto Scaling of an Autonomous AI Database	121
Enable or Disable Database In-Memory	122
Enable Ops Insights for an Autonomous AI Database	123
View Details of an Autonomous AI Database	123
Manage Customer Contacts for an Autonomous AI Database	124
Add Customer Contacts	125
Edit Customer Contacts	125
Delete Customer Contacts	125
Manage CPU or Storage Resources of an Autonomous AI Database	125
Add CPU or Storage Resources to an Autonomous AI Database	126
Remove CPU or Storage Resources from an Autonomous AI Database	127
Manage Long-Term Backups	129
Managing Primary and Standby Databases in an Autonomous Data Guard Configuration	130
Move an Autonomous AI Database to a Different Compartment	132
Restore and Recover Autonomous AI Database on Dedicated Exadata Infrastructure	133

Start, Stop, and Restart an Autonomous AI Database	134
Stop an Autonomous AI Database	134
Start an Autonomous AI Database	135
Restart an Autonomous AI Database	135
Terminate an Autonomous AI Database on Dedicated Exadata Infrastructure	136
Undelete an Autonomous AI Database	136
View Autonomous AI Database Metrics for a Database	137
View Autonomous AI Database Metrics for a Database	137
View Autonomous AI Database Metrics for a Database	137
View Autonomous AI Database Metrics for Databases in a Compartment	137
Connect to Autonomous AI Database	138
Download Client Credentials	138
Prepare for JDBC Thin Connections	139
Download, Install and Configure Oracle Instant Client	139
Prepare for TLS Walletless Connections	140
Additional Information	141
Connect Python Applications Using mTLS	142
Install Python and the python-oracledb Driver	142
Obtain Security Credentials (Oracle Wallet) and Enable Network Connectivity	144
Run Python Application with python-oracledb Thin Mode (mTLS)	145
Run Python Application with python-oracledb Thick Mode (mTLS)	147
Connect with Node.js and other Scripting Languages	150
Connect with Oracle Database Actions	152
Connect with SQL*Plus	154
Connect with Oracle SQLcl	155
Connect to Autonomous AI Database on Dedicated Exadata Infrastructure with Oracle SQL Developer	156
Connect with Oracle SQL Developer Version 18.2 or Later	157
Connect with Oracle SQL Developer Versions Earlier Than 18.2	158
View Connection Strings for an Autonomous AI Database	160
Connect to Oracle Autonomous AI Database Using JDBC with Eclipse	161
Prerequisites	161
Procedure	164
Connect to Oracle Autonomous AI Database Using JDBC with IntelliJ	167
Prerequisites	167
Procedure	170
Connect to Oracle Database Using JDBC Without a Build Tool	173
Prerequisites	174
Procedure	176
Secure Autonomous AI Database	178
Configure Zero Trust Packet Routing (ZPR) for an AVMC	178
Create and Manage Key Stores	178

Create a Key Store	178
View Key Store Details	179
Edit Key Store Details	179
Move a Key Store to Another Compartment	180
View Key Store Associated Container Database Details	180
Delete a Key Store	180
Prepare to Use Oracle Key Vault with Autonomous AI Database on Dedicated Exadata Infrastructure	181
Prerequisites	181
Create a Vault in OCI Vault Service and Add a Secret to the Vault to Store OKV REST Administrator Password	181
Create a Policy Statement for Database Service to Use Secret from OKV Vault Service	182
Update OKV Endpoint Group	182
Manage OKV Endpoints	182
Register or Deregister a Dedicated Autonomous AI Database with Data Safe	183
Rotate Encryption Keys	184
Rotate the Encryption Key of an Autonomous Container Database	184
Rotate the Encryption Key of an Autonomous AI Database	185
Set Access Control List for an Autonomous AI Database	186
Create and Manage Autonomous AI Database Software Images	187
Create an Autonomous AI Database Software Image	187
Manage Autonomous AI Database Software Images	189
View a List of Autonomous AI Database Software Images	189
View Details of an Autonomous AI Database Software Image	190
Delete an Autonomous AI Database Software Image	191
Create and Manage Elastic Pools	191
Create an Elastic Pool	191
Create an Elastic Pool Using an Existing Autonomous AI Database	191
Create an Elastic Pool While Provisioning or Cloning an Autonomous AI Database	192
Join or Leave an Elastic Pool	192
Join an Elastic Pool	192
Join an Elastic Pool While Provisioning or Cloning an Autonomous AI Database	193
Leave an Elastic Pool Autonomous AI Database	193
Manage an Elastic Pool as a Pool Leader	194
Change an Elastic Pool Shape	194
List Elastic Pool Members	194
Remove an Elastic Pool Member	195
Terminate an Elastic Pool	195
View a List of Remote Backups	195

7 Reference

Previous Feature Announcements	1
New Feature Announcements	1
December 2025	1
November 2025	2
October 2025	3
September 2025	5
August 2025	6
July 2025	7
June 2025	7
May 2025	9
April 2025	10
March 2025	11
February 2025	12
January 2025	13
New Feature Announcements (2024)	13
December 2024	14
November 2024	15
October 2024	16
September 2024	17
August 2024	18
July 2024	19
June 2024	20
May 2024	21
April 2024	23
March 2024	25
February 2024	26
January 2024	28
New Feature Announcements (2023)	28
December 2023	29
November 2023	29
October 2023	30
September 2023	32
August 2023	33
July 2023	34
June 2023	35
May 2023	36
April 2023	38
March 2023	40
February 2023	41
New Feature Announcements	42

December 2022	43
November 2022	45
October 2022	45
August 2022	46
July 2022	46
June 2022	47
May 2022	48
April 2022	49
March 2022	51
February 2022	52
January 2022	52
New Feature Announcements (2021)	53
December 2021	54
November 2021	55
October 2021	55
September 2021	56
July 2021	56
June 2021	56
May 2021	57
April 2021	57
February 2021	58
January 2021	59
New Feature Announcements (2020)	59
December 2020	60
November 2020	60
October 2020	61
September 2020	61
August 2020	61
July 2020	62
June 2020	62
May 2020	62
April 2020	63
March 2020	63
February 2020	63
January 2020	64
New Feature Announcements (2019)	64
September 2019	65
August 2019	65
July 2019	66
June 2019	66
Autonomous AI Database Supplied Package Reference	67
DBMS_CLOUD Package	67

DBMS_CLOUD for Access Management	67
DBMS_CLOUD for Objects and Files	73
DBMS_CLOUD for Bulk File Management	121
DBMS_CLOUD REST APIs	137
DBMS_CLOUD Package Format Options	145
DBMS_CLOUD Package Format Options for EXPORT_DATA	156
DBMS_CLOUD Avro, ORC, and Parquet Support	161
DBMS_CLOUD Exceptions	173
Cloud Object Storage URI Formats	173
SODA Collection Metadata on Autonomous AI Database	181
DBMS_CLOUD_REPO Package	183
DBMS_CLOUD_REPO Overview	183
Summary of DBMS_CLOUD_REPO Subprograms	186
DBMS_DCAT Package	206
About DBMS_DCAT Package	206
Connection Management Subprograms	209
CS_SESSION Package	219
SWITCH_SERVICE Procedure	220
Summary of DBMS_CLOUD_ADMIN Subprograms	221
ATTACH_FILE_SYSTEM Procedure	221
CREATE_DATABASE_LINK Procedure	223
DETACH_FILE_SYSTEM Procedure	233
DISABLE_EXTERNAL_AUTHENTICATION Procedure	234
DROP_DATABASE_LINK Procedure	235
ENABLE_EXTERNAL_AUTHENTICATION Procedure	235
DBMS_CLOUD_ADMIN Exceptions	240
DBMS_CLOUD_AI Package	240
Summary of DBMS_CLOUD_AI Subprograms	242
CREATE_PROFILE Procedure	243
Profile Attributes	244
DROP_PROFILE Procedure	248
ENABLE_PROFILE Procedure	249
DISABLE_PROFILE Procedure	249
SET_ATTRIBUTE Procedure	250
SET_PROFILE Procedure	250
GENERATE Function	251
GENERATE_SYNTHETIC_DATA Function	257
ENABLE_DATA_ACCESS Procedure	262
DISABLE_DATA_ACCESS Procedure	262
CREATE_VECTOR_INDEX Procedure	263
DROP_VECTOR_INDEX Procedure	264
DISABLE_VECTOR_INDEX Procedure	265

ENABLE_VECTOR_INDEX Procedure	266
UPDATE_VECTOR_INDEX Procedure	266
Vector Index Attributes	269
CREATE_CONVERSATION Procedure	274
CREATE_CONVERSATION Function	275
CREATE_CONVERSATION Attributes	276
UPDATE_CONVERSATION Procedure	277
SET_CONVERSATION_ID Procedure	278
GET_CONVERSATION_ID Function	279
CLEAR_CONVERSATION_ID Procedure	279
DELETE_CONVERSATION_PROMPT Procedure	280
DROP_CONVERSATION Procedure	280
FEEDBACK Procedure	281
Vector Index for FEEDBACK	283
SUMMARIZE Function	284
SUMMARIZE Parameters	285
TRANSLATE Function	286
DBMS_CLOUD_AI_AGENT Package	287
Summary of DBMS_CLOUD_AI_AGENT Subprograms	287
CREATE_AGENT Procedure	288
CREATE_AGENT Attributes	289
ENABLE_AGENT Procedure	290
DISABLE_AGENT Procedure	291
DROP_AGENT Procedure	291
CREATE_TASK Procedure	292
CREATE_TASK Attributes	294
ENABLE_TASK Procedure	295
DISABLE_TASK Procedure	295
DROP_TASK Procedure	296
CREATE_TOOL Procedure	297
CREATE_TOOL Attributes	298
ENABLE_TOOL Procedure	300
DISABLE_TOOL Procedure	300
DROP_TOOL Procedure	301
CREATE_TEAM Procedure	302
CREATE_TEAM Attributes	303
SET_TEAM Function	303
GET_TEAM Function	304
CLEAR_TEAM Function	304
RUN_TEAM Function	304
ENABLE_TEAM Procedure	305
DISABLE_TEAM Procedure	306

DROP_TEAM Procedure	306
DBMS_CLOUD_FUNCTION Package	307
CREATE_CATALOG Procedure	308
CREATE_FUNCTION Procedure	311
DROP_CATALOG Procedure	315
DROP_FUNCTION Procedure	315
LIST_FUNCTIONS Procedure	316
SYNC_FUNCTIONS Procedure	317
DBMS_CLOUD_NOTIFICATION Package	318
Summary of DBMS_CLOUD_NOTIFICATION Subprograms	320
SEND_DATA Procedure	320
SEND_MESSAGE Procedure	323
DBMS_CLOUD_PIPELINE Package	327
Prerequisites	327
Summary of DBMS_CLOUD_PIPELINE Subprograms	328
CREATE_PIPELINE Procedure	329
DROP_PIPELINE Procedure	330
RESET_PIPELINE Procedure	330
RUN_PIPELINE_ONCE Procedure	331
SET_ATTRIBUTE Procedure	332
START_PIPELINE Procedure	333
STOP_PIPELINE Procedure	333
DBMS_CLOUD_PIPELINE Attributes	334
Summary of DBMS_EXT_TABLE_CACHE Subprograms	337
ADD_BY_LIKE Procedure	337
ADD_FILE Procedure	338
ADD_LATEST_FILES Procedure	339
ADD_TABLE Procedure	340
CLEAR Procedure	341
CREATE_CACHE Procedure	342
DISABLE Procedure	343
DROP_BY_LIKE Procedure	344
DROP_CACHE Procedure	344
DROP_FILE Procedure	345
ENABLE Procedure	346
GET_USER_PROPERTY Function	346
RETIRE_FILES Procedure	347
SET_USER_PROPERTY Procedure	348
VALIDATE Procedure	349
DBMS_PIPE Package	349
DBMS_PIPE Overview for Singleton Pipes	349
Summary of DBMS_PIPE Subprograms for Singleton Pipes	350

CREATE_PIPE Function	351
RECEIVE_MESSAGE Function	353
SEND_MESSAGE Function	356
DBMS_PIPE Package	358
DBMS_PIPE Overview for Persistent Messaging Pipes	358
Summary of DBMS_PIPE Subprograms for Persistent Messaging	359
CREATE_PIPE Function	360
GET_CREDENTIAL_NAME Function	362
GET_LOCATION_URI Function	362
RECEIVE_MESSAGE Function	363
SEND_MESSAGE Function	365
SET_CREDENTIAL_NAME Procedure	367
SET_LOCATION_URI Procedure	367
Autonomous AI Database Views	368
Data Catalog Views	368
ALL_CLOUD_CATALOG_DATABASES View	369
ALL_CLOUD_CATALOG_TABLES View	370
ALL_DCAT_ASSETS View	374
ALL_DCAT_ATTRIBUTES View	375
ALL_DCAT_CONNECTIONS View	376
ALL_DCAT_ENTITIES View	376
ALL_DCAT_FOLDERS View	377
ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View	378
ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View	378
ALL_GLUE_DATABASES View	378
ALL_GLUE_TABLES View	379
DCAT_ATTRIBUTES View	380
DCAT_ENTITIES View	380
DBMS_DCAT\$SYNC_LOG View	380
DBMS_CLOUD_AI Views	381
DBA_CLOUD_AI_PROFILES View	381
USER_CLOUD_AI_PROFILES View	381
DBA_CLOUD_AI_PROFILE_ATTRIBUTES View	382
USER_CLOUD_AI_PROFILE_ATTRIBUTES View	382
DBA_CLOUD_VECTOR_INDEXES View	382
USER_CLOUD_VECTOR_INDEXES View	383
DBA_CLOUD_VECTOR_INDEX_ATTRIBUTES View	383
USER_CLOUD_VECTOR_INDEX_ATTRIBUTES View	384
USER_CLOUD_AI_CONVERSATIONS View	384
USER_CLOUD_AI_CONVERSATION_PROMPTS View	384
DBMS_CLOUD_AI_AGENT Views	385
DBA_AI_AGENTS View	385

DBA_AI_AGENTS_ATTRIBUTES View	386
USER_AI_AGENTS View	386
USER_AI_AGENTS_ATTRIBUTES View	387
DBA_AI_AGENT_TOOLS View	387
DBA_AI_AGENT_TOOL_ATTRIBUTES View	387
USER_AI_AGENT_TOOLS View	388
USER_AI_AGENT_TOOL_ATTRIBUTES View	388
DBA_AI_AGENT_TASK View	388
DBA_AI_AGENT_TASK_ATTRIBUTES View	389
USER_AI_AGENTS View	389
USER_AI_AGENT_TASK_ATTRIBUTES View	390
DBA_AI_AGENT_TEAMS View	390
DBA_AI_AGENT_TEAM_ATTRIBUTES View	390
USER_AI_AGENT_TEAMS View	391
USER_AI_AGENT_TEAM_ATTRIBUTES View	391
DBMS_CLOUD_AI_AGENT History Views	392
DBA_AI_AGENT_TEAM_HISTORY View	392
USER_AI_AGENT_TEAM_HISTORY View	393
DBA_AI_AGENT_TASK_HISTORY View	394
USER_AI_AGENT_TASK_HISTORY View	394
DBA_AI_AGENT_TOOL_HISTORY View	395
USER_AI_AGENT_TOOL_HISTORY View	396
DBMS_CLOUD_FUNCTION Views	397
DBA_CLOUD_FUNCTION View	397
DBA_CLOUD_FUNCTION_CATALOG View	397
USER_CLOUD_FUNCTION View	397
USER_CLOUD_FUNCTION_CATALOG View	398
USER_CLOUD_FUNCTION_ERRORS View	398
Pipeline Views	398
DBA_CLOUD_PIPELINES View	398
USER_CLOUD_PIPELINES View	399
DBA_CLOUD_PIPELINE_HISTORY View	399
USER_CLOUD_PIPELINE_HISTORY View	400
DBA_CLOUD_PIPELINE_ATTRIBUTES View	401
USER_CLOUD_PIPELINE_ATTRIBUTES View	401
Track DBMS_CLOUD Load Operations	401
DBA_LOAD_OPERATIONS and USER_LOAD_OPERATIONS Views	401
Events for Autonomous AI Database on Dedicated Exadata Infrastructure	402
Exadata Infrastructure Event Types	403
Cloud Exadata Infrastructure Event Types	403
Autonomous Exadata VM Cluster Event Types	403
Cloud Autonomous Exadata VM Cluster Event Types	407

Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types	409
Autonomous Container Database Event Types	411
Autonomous AI Database Event Types	416
Availability Service Level Agreements (SLAs)	421
API Reference for Autonomous AI Database on Dedicated Exadata Infrastructure	423
API to Manage Exadata Infrastructure Resources	423
API to Manage Autonomous Exadata VM Clusters	425
API to Manage Autonomous Container Databases	426
API to Manage Autonomous AI Databases	427
API to Manage Autonomous Data Guard Configuration	428
API to Manage Autonomous Virtual Machines	429
API to Manage Autonomous AI Database Software Images	430
Troubleshooting Autonomous AI Database on Dedicated Exadata Infrastructure	430
Unable to Access Master Encryption Keys	430
Unable to Access Vault	431
Backup to Network File System (NFS) is Failing	432
Unable to Mount Network File System (NFS)	432
Unable to write a file in Network File System (NFS)	433
Unable to get outbound traffic from APEX	433
Backup Retention Issues with ZDLRA	434
Delay in Connecting to Autonomous AI Database with SQL*Plus	434
Certify Applications for Compatibility Between Autonomous AI Database Serverless and Dedicated	435
Feature Parity Between Oracle Public Cloud and Exadata Cloud@Customer	437
Characteristics of Infrastructure Shapes	438
Exadata Base Systems	439
Exadata X11M Systems	439
Exadata X10M Systems	444
Exadata X9M-2 Systems	445
Exadata X8M-2 Systems	446
Exadata X8-2 Systems	447
Exadata X7-2 Systems	448
CPU Overprovisioning with Autonomous AI Database on Dedicated Exadata Infrastructure	449
About CPU Overprovisioning	449
Implementing CPU Overprovisioning	449
Known Issues	450

1

Overview

New Feature Announcements

Announcements for the noteworthy changes made to Autonomous AI Database on Dedicated Exadata Infrastructure.

April 2026

Feature	Description
Additional log support for OCI Logging Service	In addition to the available logs on Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, the following logs are available to be used with OCI Logging Service Unified audit logs for capturing the most accurate record of any database activity See Observe Resource Logs with OCI Logging Service for more details.
Provision an Autonomous Container Database with 19c or 26ai database software version in the same Autonomous Exadata VM Cluster.	For all new Autonomous Exadata VM Clusters (AVMCs) provisioned on Autonomous AI Database on Dedicated Exadata Infrastructure after April 7 2026, users will be able to provision both 19c and 26ai database software versions in the same AVMC. For existing AVMCs with 26ai or 23ai tags, you can now provision an Autonomous Container Database with 19c or 26ai database software versions in the same AVMC. See Create an Autonomous Exadata VM Cluster and Create an Autonomous Container Database for instructions.

March 2026

Availability of new Dynamic Performance Views on Oracle Autonomous AI Database on Dedicated Exadata Infrastructure	You can use Dynamic Performance Views to: - View History of statistical summary of system performance metrics - View Historical information for Exadata cell See Database Observability with Dynamic Performance Views for additional information.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@Azure	Using Oracle Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@Azure, you can run Oracle Database workloads on Oracle Cloud Infrastructure (OCI) managed Exadata Infrastructure within Azure public regions and Availability Zones. See Autonomous AI Database on Dedicated Exadata Infrastructure in Azure Region for details.
DBMS_CLOUD_FUNCTION Package, with support for External Procedures	With Oracle Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can use the subprograms in DBMS_CLOUD_FUNCTION package to develop applications with SQL and PL/SQL User Defined Functions that invoke remote functions deployed in the OCI and AWS cloud, and external procedures implemented in a custom OCI compute VM. Cloud Functions are serverless compute engines in the OCI and AWS cloud. External procedures are routines written in a third-generation language and invoked as SQL functions or PL/SQL procedures or functions. SQL and PL/SQL are best suited for fast and efficient data and transaction processing in the database. Cloud functions and External procedures can complement SQL and PL/SQL in-database processing by executing compute and memory-intensive tasks in a remote VM and providing the results back to the database. Typical examples of such tasks are scientific and engineering problems whose computational libraries exist in the cloud, or in the Linux OS (and are not easily portable to the database), offline data analysis, control of real-time devices and processes, and so on. See Invoke Cloud Functions and External Procedures and DBMS_CLOUD_FUNCTION Package for details.
Integrate generative AI capabilities into Python workflows	On Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, you can now use Select AI for Python to access DBMS_CLOUD_AI capabilities in Autonomous AI Database from Python applications. With Select AI for Python, you can: • Ask questions about your database data using natural language (NL2SQL) • View and explain AI-generated SQL queries • Chat with your LLM with customized prompts • Use automated processes for creating vector indexes and using Retrieval-Augmented Generation (RAG) • Generate synthetic data using large language models to populate database tables • Conduct conversational AI sessions for short-term and long-term memory See Select AI for Python for more information.

February 2026

Feature	Description
Improved visibility into ASM rebalance progress during storage scaling operations for Exadata VM Clusters.	In Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, you can view detailed ASM rebalance progress when scaling storage. The Scale VM Cluster option is available even while storage attachment is in progress, and the cluster is in the UPDATING state. The clusters where storage attachment is complete and rebalance is running will transition back to AVAILABLE, so you can start new operations without waiting for all clusters to finish. The VM Clusters automatically return to AVAILABLE after storage attachment finishes. You can also see clearer progress for storage attachment and rebalance, including completion status, progress percentage, and relevant log messages.
Use DB_NOTIFICATIONS view to see information on maintenance status notifications.	On Oracle Public Cloud deployments, you can use DB_NOTIFICATIONS view to see information on maintenance status notifications. See View Maintenance Status Notifications for more details.
DBMS_DCAT Package Support	With Oracle Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can use functions and procedures in the DBMS_DCAT package to help Autonomous AI Database users leverage the data discovery and centralized metadata management system of OCI Data Catalog. DBMS_DCAT procedures and functions connect Autonomous AI Database to Data Catalog and then synchronize the assets with the database, creating protected schemas and external tables. You can then query object store using those external tables, easily joining external data with data stored in Autonomous AI Database. See DBMS_DCAT Package for more details.
Ability to choose Autonomous Recovery Service as the backup destination in Oracle Database@AWS.	In Autonomous AI Database on Oracle Database@AWS, you can choose Autonomous Recovery Service as the backup destination while creating the ACD. This is the recommended option for the backup destination. See Create an Autonomous Container Database and Edit Autonomous Container Database Backup Settings for more details.
Summarize text with Select AI	Select AI on Autonomous AI Database on Dedicated Exadata Infrastructure enables you to generate a summary of your text directly in a prompt or from a URI. You can customize how the summary is generated. See See Generate a Summary with Select AI, Example: Select AI Summarize, GENERATE Function, and SUMMARIZE Function to explore this feature.

Feature	Description
Translate text using Select AI	The Translate feature of Select AI on Autonomous AI Database on Dedicated Exadata Infrastructure enables you to translate your text across multiple languages using SQL or PL/SQL, as supported by your AI provider. See Translate Text with Select AI, Example: Select AI Translate, GENERATE Function, and TRANSLATE Function to explore this feature
Interact with metadata to improve Select AI SQL query generation	Select AI on Autonomous AI Database on Dedicated Exadata Infrastructure includes the following new features: • Select AI enables you to add table comments to the metadata of the LLM for a better SQL generation. You can achieve this by specifying comments to true in your AI profile. See "Example: Improve SQL Generation with Table and Column Comments" in Select AI Examples. • Select AI integrates annotations in Oracle AI Database 26ai by adding it to the metadata that is sent to the LLM. You can achieve this by specifying annotations to true in your AI profile. See "Example: Improve SQL Generation with Table and Column Annotations" in Select AI Examples. • Select AI adds foreign key and referential key constraints into the metadata of the LLM to generate accurate JOIN conditions. You can achieve this by specifying constraints to true in your AI profile. See "Example: Improve SQL Query Generation with Foreign Key and Referential Key Constraints" in Select AI Examples.
Provide Feedback to Improve LLMs with Select AI	On Autonomous AI Database on Dedicated Exadata Infrastructure, Select AI enables you to give feedback on generated NL2SQL results to improve query generation accuracy. Feedback can be given on the SQL command line as well as through PL/SQL procedures. See Provide Feedback to Improve LLMs, Example: Select AI Feedback, and FEEDBACK Procedure to explore the feature.
Create and Customize Conversations with Select AI	On Autonomous AI Database on Dedicated Exadata Infrastructure, Select AI supports customizable conversations enabling you to use Select AI with different topics without mixing context, improving both flexibility and efficiency. You can create, set, delete, and update conversations through the DBMS_CLOUD_AI conversation procedures and functions. See Use and Customize Conversations and Example: Enable Conversations in Select AI, and Summary of DBMS_CLOUD_AI Subprograms for more information.

Feature	Description
Generate Property Graph Data with Select AI	On Autonomous AI Database on Dedicated Exadata Infrastructure, you can use Select AI to graph structures and query SQL property graphs using natural language, reducing the time and effort in building graph queries using the GRAPH_TABLE operator. See Query Graph Data with Property Graphs, Example: Select AI for Property Graphs, and Example: Sample Prompts for Property Graphs for more information.
Select AI Agent (autonomous agent framework)	Select AI Agent (autonomous agent framework) introduces a framework for building agentic workflows enabling developers to create agents, tools, tasks, and teams that reason, act, and collaborate using generative AI within the database. Agents can plan actions, use built-in or custom tools (such as SQL, RAG, Websearch, or Notification), and maintain context across multi-turn conversations. This feature enables developers to create agentic workflows using SQL and PL/SQL that support AI-driven automation, data interaction, and human-in-the-loop control. See Build Autonomous Agents with Select AI Agent, DBMS_CLOUD_AI_AGENT Package, DBMS_CLOUD_AI_AGENT Views, and Examples of Using Select AI Agent to explore this feature.

January 2026

Feature	Description
Configure Autonomous Data Guard between ACD in OCI and ACD in AWS region.	You can create and manage Autonomous Data Guard between an ACD in OCI and an ACD in AWS. This feature provides full Autonomous Data Guard functionality for disaster recovery and high availability across clouds. See Manage Autonomous Data Guard Configuration for more details.
Use AWS Key Management Service (AWS KMS) to manage Master Encryption Keys in Autonomous AI Database on Oracle Database@AWS.	In Autonomous AI Database on Oracle Database@AWS, you can use AWS Key Management Service (AWS KMS) to manage Master Encryption Keys in AWS. See Autonomous AI Database on Dedicated Exadata Infrastructure in AWS Region for more details.
Ability to choose Autonomous Recovery Service as the backup destination.	On Oracle Public Cloud deployments, you can choose Autonomous Recovery Service as the backup destination while creating the ACD. This is the recommended option for the backup destination. See Create an Autonomous Container Database and Edit Autonomous Container Database Backup Settings for more details.

Feature	Description
Ability to use a third network interface card (NIC) for specific database operations like Autonomous Data Guard and cloning the Autonomous AI Database.	On Exadata Cloud@Customer deployments, if you have enabled a Disaster Recovery Network, you can use the third NIC that you have configured for specific database operations like Autonomous Data Guard (AuDG) and cloning the Autonomous AI Database. The Data guard and cloning operations are separated from regular network traffic, reducing the risk of network congestion and improving performance. To use the third NIC with your Autonomous Virtual Machine Cluster (AVMC), you must first submit a Service Request (SR) to Oracle Support. This Service Request must be made before the AVMC is created. Once enabled, all the Data Guard and cloning operations from database instance uses only the 3rd NIC. See Network Requirements for Oracle Exadata Database Service on Cloud@Customer for more details.

About Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database provides an easy-to-use, fully autonomous database that scales elastically, delivers fast query performance, and requires no database administration.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure is a highly automated, fully managed database environment running in Oracle Cloud Infrastructure (OCI) with committed hardware and software resources. These isolated resources enable organizations to meet stringent security, availability, and performance requirements while reducing cost and complexity.

You can configure your database in two different modes based on the workload type, as:

- Autonomous AI Lakehouse
- Autonomous AI Transaction Processing

Autonomous AI Lakehouse

Oracle Autonomous AI Lakehouse is a cloud data warehouse service that eliminates virtually all the complexities of operating a data warehouse, securing data, and developing data-driven applications.

The service automates provisioning, configuring, securing, tuning, scaling, patching, backing up, and repairing of the data warehouse. Unlike other fully managed cloud data warehouse solutions that only patch and update the service, Autonomous AI Lakehouse features elastic, automated scaling, performance tuning, security, and a broad set of built-in converged database capabilities that enable simpler queries across multiple data types, machine learning analysis, simple data loading, and data visualizations.

Autonomous AI Lakehouse service is available in both the Oracle public cloud and your own data center on Exadata Cloud@Customer.

Autonomous AI Transaction Processing

Oracle Autonomous AI Transaction Processing is a cloud database service that eliminates the complexity of operating and securing high-performance databases.

The service automates provisioning, configuring, tuning, scaling, patching, encrypting, and repairing of databases. Additionally, the service includes all of Oracle's advanced database options, such as real application clusters (RAC), multitenant, partitioning, in-memory, advanced security, and advanced compression. The service is built to support everything from simple web apps to large and sophisticated applications that are critical for business operation.

Autonomous AI Transaction Processing service is available in both the Oracle public cloud and your own data center on Exadata Cloud@Customer.

Key Features of Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database on Dedicated Exadata Infrastructure provides the following set of features to offer affordable, feature-rich database service in the cloud.

- **Fully Managed:** Oracle simplifies end-to-end management of the database with self-driving Autonomous AI Database technology. With Autonomous AI Database, you do not need to configure or manage any hardware or install any software. Autonomous AI Database handles creating the database, backing up the database, patching and upgrading the database, growing or shrinking the database, and other routine database maintenance tasks without human intervention and with no service downtime.
- **Fully Tuned:** Autonomous AI Database is designed as a "Load and Go" service to define tables, load data, and start running queries. Autonomous AI Database provides great performance out of the box. You can run your queries using Oracle database Client Drivers from all popular programming languages, any business analytics tool, cloud service, built-in SQL worksheet, or even a machine-learning notebook.
- **Fully Elastic:** Autonomous AI Database is a fully elastic service. When you get started with Autonomous AI Database, you can simply specify the number of CPUs (a unit expression of Compute cores) and the database's storage capacity. Any time later, you can either enable an auto-scaling feature or manually scale, increase or decrease, either the CPUs or the storage capacity. When you enable auto-scaling or perform manual scaling, your database resources shrink or grow without any downtime or service interruptions. You can size the Autonomous AI Database to the exact compute and storage required or even shut off idle compute to save money.
- **Highly Available:** Autonomous AI Database provides a highly available database with features such as:
 - Autonomous Data Guard that can be used to protect critical databases from failures and disasters.
 - Automatic and manual backups that can be used to restore and recover your database to any point-in-time within the retention period.
- **High Performing:** Autonomous AI Database provides a high performing database with features such as:
 - Automated index management
 - Automatic real-time statistics gathering
 - Optimized high-frequency single-row inserts for applications

- preconfigured resource profiles for different types of users to achieve optimized query performance and concurrent workloads
- **Easy to Use:** Autonomous AI Database includes a complete REST based API and a cloud-based service console for managing the service (for tasks such as stopping, starting, or scaling the service) and monitoring the service (for functions such as viewing the recent activity levels on the data warehouse).
- **Compatible with Existing Applications and Tools:** Autonomous AI Database is built upon Oracle Database so that the applications and tools that support Oracle Database also support Autonomous AI Database. These tools and applications can either be in your data center or a public cloud and connect to Autonomous AI Database using standard SQL*Net connections.
- **Integration with Third Party Tools and Utilities:** With Autonomous AI Database, you can integrate your database applications with other Oracle Cloud services such as Oracle Analytics Cloud, Oracle GoldenGate Cloud Service, and Oracle Data Integration Platform Cloud. You can use the Database Migration Utility provided by Autonomous AI Database to easily migrate your data from MySQL, PostgreSQL, SQL Server, and other databases. Autonomous AI Database supports connectivity via SQL*Net, JDBC, and ODBC.
- **Enriched with Built-in Tools:** Autonomous AI Database comes with built-in tools such as:
 - **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 Database Actions**, a browser-based interface of Oracle SQL Developer.

Deployment Choices for Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database on Dedicated Exadata Infrastructure offers three deployment possibilities for operational control and isolation:

- **Oracle Cloud Infrastructure:** A private dedicated database within a public cloud that completely isolates your data and operations. Autonomous AI Databases on Oracle Public Cloud get dedicated system resources such as processor, memory, network, or storage to offer greater control over operational policies and runtime customizations.

With the *Dedicated Region Cloud@Customer (DRCC)* option, customers get an entire OCI region as it exists in Oracle's Public Cloud, deployed inside the customer data center. Autonomous AI Databases deployed in DRCC are functionally the same as those deployed in Oracle Public Cloud regions, except that they get an OCI region inside customer data centers.

- **Exadata Cloud@Customer Infrastructure:** An Autonomous AI Database in *your data center* to meet regulatory, data sovereignty, or network latency requirements for workloads that cannot move to the Public Cloud. This deployment option enables IT to easily deliver self-service databases to business users and developers while ensuring the security and governance of all data.
- **Multicloud :** An Oracle Autonomous AI Database on Dedicated Exadata Infrastructure deployed in a multicloud region. These include AWS cloud regions and Azure cloud regions. This deployment option enables applications running within AWS or Azure that require high performance and low latency to benefit from a fully managed Oracle Database that is highly available, scalable, secure, and integrated with many operational services. All

features supported on Oracle Public Cloud will also apply to Multicloud deployments, except for the features listed in [Unsupported Features in Oracle Database@AWS](#) and [Unsupported Features in Oracle Database@Azure](#).

All the platforms support a standard set of features for Autonomous AI Databases with a few differences. All through this documentation, you will encounter the following badges as applicable:

- APPLIES TO:  Oracle Public Cloud only
- APPLIES TO:  Exadata Cloud@Customer only

For a glance at the features supported exclusively by Oracle Public Cloud or Exadata Cloud@Customer deployment choices, refer to [Feature Parity Between Oracle Cloud and Exadata Cloud at Customer](#).

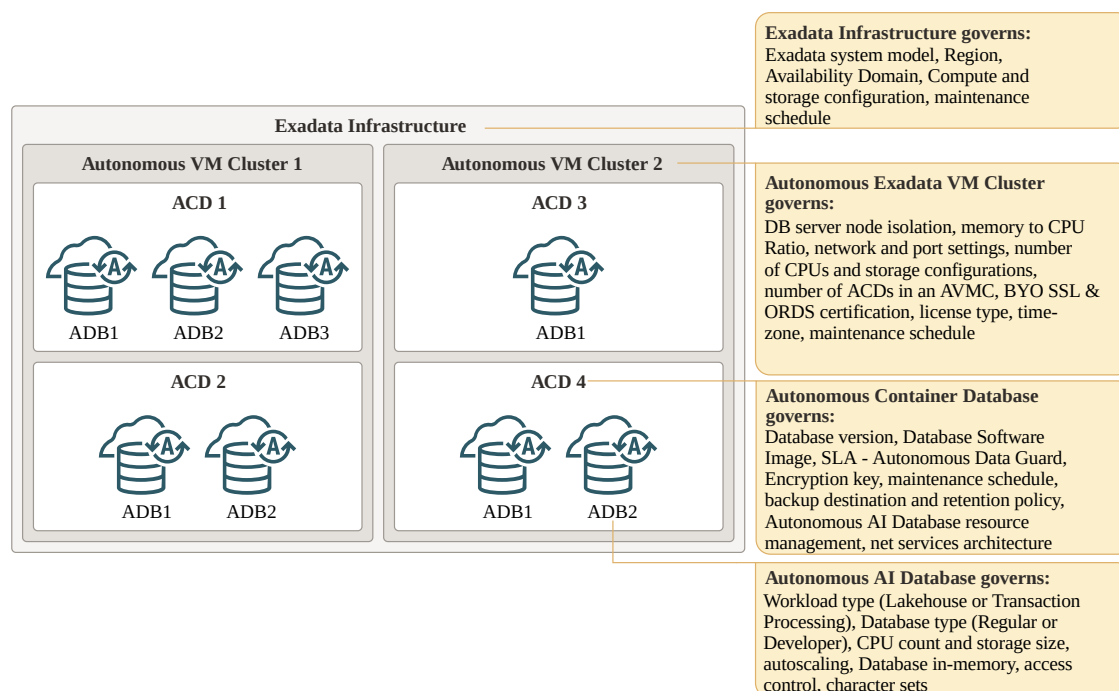
Components of Autonomous AI Database on Dedicated Exadata Infrastructure

The following diagram illustrates the infrastructure components of an Autonomous AI Database on Dedicated Exadata Infrastructure.



Tip

In the following image, you can click the component that you want to explore further.



 Tip

For a practical use case that demonstrates how to leverage these components to develop, test, and deploy a database application, refer to [Configure Autonomous AI Database on Dedicated Exadata Infrastructure](#).

Autonomous AI Database on Dedicated Exadata Infrastructure is based upon these kinds of Oracle Cloud resources:

- An **Exadata Infrastructure** resource represents the Exadata Database Machine system in your data center, together with the networking configuration that connects it to Oracle Cloud. For an overview of Exadata Infrastructure, its features, and how to use them, see [About Exadata Infrastructure](#).
- An **Autonomous Exadata VM Cluster (AVMC)** resource provides the link between the Exadata Infrastructure resource and the Autonomous Container Database resources in your deployment. It is a set of symmetrical VMs (virtual machines) across all compute nodes of the underlying Exadata Infrastructure resource. You can:
 - Create multiple AVMCs on a single Exadata Infrastructure resource. Support for multiple VM Clusters lets you:
 - * Schedule separate maintenance runs for each AVMC on the same Exadata Infrastructure.
 - * Choose different license models for Autonomous AI Databases on the same Exadata Infrastructure.
 - * Create and test Autonomous Data Guard between AVMCs on the same Exadata Infrastructure.
 - * Customize compute, storage, and memory of each AVMC configuration for the intended workload.
 - Create both Autonomous Exadata VM Clusters and Exadata Database VM Clusters on a common Exadata Infrastructure resource.

For an overview of AVMCs, their features, and how to use them, see [About Autonomous Exadata VM Cluster](#).

 Note

You can only create a Multiple VM - Autonomous VM Cluster (AVMC) on Exadata Infrastructure resources created in Oracle Cloud after the launch of the multiple VM Autonomous AI Databases feature. Please create a service request in My Oracle Support if you need to address this limitation and want to add an Autonomous VM Cluster to older Exadata Infrastructure resources. For help with filing a support request, see [Create a Service Request in My Oracle Support](#).

- An **Autonomous Container Database (ACD)** resource provides a container for your Autonomous AI Databases. You can create multiple ACDs in a single AVMC resource, but you must create at least one before you can create any Autonomous AI Databases. For an overview of ACDs, their features, and how to use them, see [About Autonomous Container Database](#).

ACDs can be provisioned with *Oracle Database 19c* or *Oracle Database 26ai*, which applies to all the Autonomous AI Databases provisioned in it. However, to provision an ACD with 26ai as its Database software version, the parent AVMC must be *ECPU-based*

with the *DatabaseVersion* tag set to 26ai. For a comprehensive list of database features supported by each of these database versions, see Oracle Database 26ai or [Oracle Database 19c](#) on Oracle Help Center.

- An **Autonomous AI Database** resource is a user database. You can create many Autonomous AI Databases in a single Autonomous Container Database resource. For an overview of Autonomous AI Databases, their features, and how to use them, see About Autonomous AI Database.
- An **Autonomous Backup** resource is a backup of an Autonomous AI Database. Oracle creates these resources for you automatically. See Backup and Restore Autonomous Database for more details.

User Roles Associated with Autonomous AI Database on Dedicated Exadata Infrastructure

The tasks involved in setting up and using the Autonomous AI Database on Dedicated Exadata Infrastructure can be grouped into three logical roles: a few people act as fleet administrators, more act as database administrators, and even more act as database users. A given person can assume one or more of these roles, depending on how you decide to isolate and distribute duties.

- **Fleet Administrator.** Fleet administrators create, monitor and manage the core (Exadata Infrastructure and Autonomous Exadata VM Cluster) and Autonomous Container Database resources.

To perform these duties, a fleet administrator must be an Oracle Cloud user whose permissions permit the management of these resources and permit the use of the networking resources that need to be specified when creating these resources.

- **Database Administrator.** Database administrators create, monitor and manage Autonomous AI Databases. Additionally, they create and manage Oracle AI Database users within these databases, and provide others the information necessary access the database.

To perform these duties, a database administrator must be an Oracle Cloud user whose permissions permit the management of Autonomous AI Database and Autonomous Backup resources and permit the use of the Autonomous Container Database and networking resources that need to be specified when creating an Autonomous AI Database. When creating an Autonomous AI Database resource, the database administrator defines and gains access to the ADMIN administrative user account for the database.

- **Database User.** Database users are the developers who write applications that connect to and use an Autonomous AI Database to store and access the data. Database users do not need Oracle Cloud accounts: they gain network connectivity to and connection authorization information for the database from the database administrator.

Autonomous AI Database Components

About Exadata Infrastructure

The Exadata Infrastructure (EI) resource represents the Exadata Database Machine system, together with the networking configuration that connects it to Oracle Cloud. It hosts the underlying components Autonomous VM Clusters, Autonomous Container Databases, and Autonomous AI Databases.

EI is one of the four components of the four-level database architecture model, which is the foundation for an Autonomous AI Database on Dedicated Exadata Infrastructure.

To get a holistic idea of the four layer architecture used with Autonomous AI Database on Dedicated Exadata Infrastructure and understand Exadata Infrastructure's positioning in this architecture, refer to Components of Autonomous AI Database on Dedicated Exadata Infrastructure.

EI is created by Fleet Administrators. See User Roles Associated with Autonomous AI Database on Dedicated Exadata Infrastructure to learn more.

The fleet administrator must first create an EI resource to host the Autonomous AI Database resources following the instructions in Create an Exadata Infrastructure Resource. Depending on the platform that you are deploying on, the following pages can be helpful in deciding the EI attributes such as the Infrastructure shape and capacity.

- Deploying on Oracle Public Cloud:
 - Characteristics of Infrastructure Shapes
 - Resource Limits
- Deploying on Exadata Cloud@Customer:
 - Preparing for Exadata Cloud@Customer
 - Provisioning Exadata Cloud@Customer Systems
- Deploying on Multicloud:
 - [Service Limits for Oracle Database@AWS](#)
 - [Service Limits for Oracle Database@Azure](#)

Note

You can skip creating a new EI resource if you are an existing customer for Oracle Exadata Database Service on Dedicated Infrastructure or Oracle Exadata Database Service on Cloud@Customer and already provisioned an EI resource. You can deploy both Autonomous AI Database and Exadata Database Service on the same EI.

Exadata Infrastructure resources emit events, which are structured messages that indicate changes in resources. Refer to Cloud Exadata Infrastructure Event Types for a list of event types that Exadata Infrastructure instances emit on Oracle Public Cloud and Exadata Infrastructure Event Types for listings of the operations that generate events for Autonomous AI Database on Exadata Cloud@Customer.

Exadata Infrastructure Requirements

Note

Depending on your Cloud Service Provider, see the following prerequisites for Multicloud deployments:

- [Prerequisites for Oracle Database@AWS](#)
- [Prerequisites for Oracle Database@Azure](#)

IAM Policy Requirements

You must have an Oracle Cloud Infrastructure account with privileges granted through required IAM Policies. The required policies depend on the operation you are performing. For a list of IAM policies pertaining to Exadata Infrastructure, see Policies to Manage Exadata Infrastructure Resources.

Minimum Resource Requirements

Ensure your service limits show at least one supported Exadata system shape available. Before attempting to create the infrastructure resources, request a service limit increase if necessary. See Request Service Limit Increase for guidance.

Note

You need to have sufficient remaining resources on the EI to support the creation of at least one Autonomous Exadata VM Cluster.

Database Features Managed from Exadata Infrastructure

The following features of Autonomous AI Database can be defined and managed at the Exadata Infrastructure (EI) level.

Feature	Notes	Further Reference
Region and Availability Domain When you create an EI resource, you must select a region and availability domain. A region is a localized geographic area, and an availability domain is one or more data centers located within a region. A region is composed of one or more availability domains.	Regions are grouped into realms. Your tenancy exists in a single realm and can access all regions that belong to that realm. You cannot access regions that are not in your realm. Ensure that you select the most appropriate region for your EI as it cannot be changed later.	Regions and Availability Domains
Exadata System Model and Shape Exadata Infrastructure can be provisioned on different Exadata System models. Each model comes in different shapes equipped with a fixed amount of memory, storage, and network resources.	The total resources allocated to your Autonomous AI Database on Dedicated Exadata Infrastructure is determined by the Exadata System (and shape) used to provision your service. The available Exadata System Shapes depends on your choice of Exadata System model.	Exadata System Shapes Characteristics of Infrastructure Shapes

Feature	Notes	Further Reference
Compute and Storage Configuration You can specify the number of database and storage servers for configuring your EI resource.	• On Oracle Public Cloud and Multicloud: For X8M-2 and X9M-2 systems, the default configuration is 2 database servers and 3 storage servers. However, you have the flexibility to choose a different value for database servers and storage servers within a valid range. The acceptable range of values is 2 to 32 for database servers and 3 to 64 for storage servers. • On Exadata Cloud@Customer deployments: The number of database servers and storage servers is pre-determined based on the Exadata system shape chosen above. You can not select a different value for database servers, but you can select another value for the Storage servers as per your requirement. In the case of X10M systems, you can pick the number of database servers and storage servers within a valid range, depending on the X10M shape you chose above.	
Maintenance Schedule In general, Oracle schedules and performs entire fleet maintenance spread throughout each quarter and monthly infrastructure security fixes for vulnerabilities with CVSS scores greater than or equal to 7. You can let Oracle handle maintenance scheduling, or you can set a specific maintenance window when Oracle can begin maintenance operations.	You can define or modify the maintenance schedule settings for an EI to be managed by Oracle or you can set a custom maintenance schedule.	Settings in Maintenance Schedule that are Customizable Management Operations on a Scheduled Maintenance
Contact Email You can provide contact emails where you can receive operational notifications, announcements, and unplanned maintenance notifications regarding your Exadata Infrastructure.	Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.	

Exadata Infrastructure Management Operations

You can perform the following management operations on an Exadata Infrastructure.

Operation	Task Instructions
Create an Exadata Infrastructure Resource	For Oracle Public Cloud deployments, see Create an Exadata Infrastructure Resource For Multicloud deployments, see Create an Exadata Infrastructure Resource in AWS region
Change the Maintenance Schedule of an Exadata Infrastructure Resource	Change the Maintenance Schedule of an Exadata Infrastructure Resource
Move an Exadata Infrastructure Resource to a Different Compartment	APPLIES TO: Oracle Public Cloud APPLIES TO: Exadata Cloud@Customer Move an Exadata Infrastructure Resource to a Different Compartment
View Details of an Exadata Infrastructure Resource	View Details of an Exadata Infrastructure Resource
View a List of Exadata Infrastructure Resources	View a List of Exadata Infrastructure Resources
View and Manage Scheduled Maintenance of an Exadata Infrastructure Resource	View and Manage Scheduled Maintenance of an Exadata Infrastructure Resource
View Past Maintenance of an Exadata Infrastructure Resource	View Past Maintenance of an Exadata Infrastructure Resource
Terminate an Exadata Infrastructure Resource	Terminate an Exadata Infrastructure Resource

Note

For Multicloud deployments, an Exadata Infrastructure Resource can also be terminated from the AWS console. See [Terminate an Exadata Infrastructure Resource from the AWS console](#).

Manage Customer Contacts for an Exadata Infrastructure	Manage Customer contacts for an Exadata Infrastructure

The above listed operations can also be achieved using API. See [API to Manage Exadata Infrastructure Resources](#) for further reference.

About Autonomous Exadata VM Cluster

Autonomous VM Clusters (AVMCs) allow a physical Exadata Cluster (Machine) to be partitioned into multiple virtual clusters. They can be used to isolate environments for different database workloads through separate access rules, network configurations, as well as customizable compute memory, and storage resources.

The AVMC is one of the infrastructure components of the four-level database architecture model on which an Autonomous Container Database is built. One or more AVMCs are

provisioned inside an Exadata Infrastructure resource (EI), providing the link between the EI and the Autonomous Container Database resources in your deployment.

AVMCs provide isolated environments for different database workloads through separate access rules, network configurations, as well as customizable compute memory, and storage resources.

To get a comprehensive idea of the four layer architecture used with Autonomous AI Database on Dedicated Exadata Infrastructure and understand AVMC's positioning in this architecture, refer to Components of Autonomous AI Database on Dedicated Exadata Infrastructure.

AVMCs are created, monitored, and managed by Fleet Administrators. See User Roles Associated with Autonomous AI Database on Dedicated Exadata Infrastructure to learn more.

Autonomous Exadata VM Cluster Requirements

Note

26ai databases can be created only on Autonomous Exadata VM Clusters (AVMCs) provisioned with the required Tags. For the tag requirements, see 26ai Database Software Version Tag Requirements. However, Multicloud deployments do not require a special tag and will automatically support both 19c and 26ai databases.

IAM Policy Requirements

Deployment Choice	IAM Policies
Oracle Public Cloud and Multicloud	manage cloud-autonomous-vmclusters use cloud-exadata-infrastructures
Exadata Cloud@Customer	manage autonomous-vmclusters use exadata-infrastructures

Minimum Resource Requirements

To create one *Autonomous Exadata VM Cluster*, you need **at least**:

- 40 ECPU's per node
- 120GB memory per node
- 338.5GB local storage per node
- 6.61TB Exadata storage

There are network requirements to be met before provisioning an Autonomous Exadata VM Cluster. See Create an Autonomous Exadata VM Cluster for the full requirements.

Note

You must provision an Exadata Infrastructure resource before creating an AVMC. See Create an Exadata Infrastructure Resource for instructions.

Database Features Managed from Autonomous Exadata VM Cluster

The following features of Autonomous AI Database can be defined and managed at the Autonomous Exadata VM Cluster (AVMC) level.

Autonomous AI Database Feature	Notes	Further Reference
Compute Model This is the compute billing metric for Autonomous AI Database.	Starting May 28th, 2025 , Autonomous AI Database on Dedicated Exadata Infrastructure does not allow you to create new AVMCs using the OCPU billing metric. All the new AVMCs can only be provisioned with ECPUs. Existing OCPU AVMCs and Autonomous AI Databases will continue to operate as usual. You can have your OCPU AVMCs and respective Autonomous AI Databases updated to ECPU via service requests. See Doc ID 2998755.1 for more information.	Compute Models
 Note Multi cloud deployments support ECPU compute model only.		
DB Server Selection AVMCs can be provisioned across any number of DB servers (VMs). You can view and select from a list of available VMs during provisioning of an AVMC.	During provisioning of an AVMC, you must select at least 2 DB Servers to deploy an AVMC resource. A minimum of 2 DB Servers are needed for high availability (HA) configuration. You can not add or remove DB Servers after provisioning an AVMC.	

Autonomous AI Database Feature	Notes	Further Reference
Maintenance Schedule In general, Oracle schedules and performs entire fleet maintenance spread throughout each quarter. You can let Oracle handle maintenance scheduling, or you can set a specific maintenance window when Oracle can begin maintenance operations.	You can schedule separate maintenance runs for each Autonomous VM Cluster on the same Exadata Infrastructure. You can change the maintenance schedule from the default (No preference, which permits Oracle to schedule maintenance as needed) by selecting the months, weeks, days and hours for the schedule. You can also set a lead time to receive a notification message about an upcoming maintenance from Oracle.	Specify When Maintenance Can Occur
Resource Management You can customize compute, storage, and memory of each Autonomous Exadata VM Cluster configuration for the intended workload. You can monitor these resources using by viewing resource allocation and usage visualizations.	Optionally, while provisioning an AVMC, you can define a suitable value for the following resource management attributes to suit your needs: • Maximum number of Autonomous Container Databases: The number of ACDs specified represents the upper limit on ACDs. • CPU count per VM or node: Specify the CPU count for each individual VM. The minimum value is 20 ECPU's per VM. • Database memory per CPU (GB): The memory per CPU allocated for the Autonomous AI Databases in the Autonomous VM Cluster. • Allocate Storage for Local Backups: For Exadata Cloud@Customer, you can check this option to configure the Exadata storage to enable local database backups. • Database storage (TB): Data storage allocated for Autonomous AI Database creation in the Autonomous VM Cluster. The terms VM and node are used interchangeably between Oracle Exadata Cloud@Customer and Oracle Public Cloud deployments.	Resource Usage Visualizations

Autonomous AI Database Feature	Notes	Further Reference
Network Settings On Oracle Public Cloud and Multicloud deployments, you define network access control using components of the Networking service. You create a Virtual Cloud Network (VCN) containing private Subnets in which your Autonomous AI Database are network-accessible.	APPLIES TO:  Oracle Public Cloud only The following settings can be adjusted while provisioning an AVMC: • Virtual cloud network: The virtual cloud network (VCN) in which you want to create the new AVMC. • Subnet: A subnet within the above selected VCN for the new Autonomous Exadata VM Cluster.	
VM cluster network On Exadata Cloud@Customer, you define network access controls by specifying a client network within your data center and recording it in a VM Cluster Network resource within the Exadata Infrastructure resource.	APPLIES TO:  Exadata Cloud@Customer only The VM cluster network in which to create the new Autonomous Exadata VM Cluster is selected during while provisioning an AVMC.	Preparing for Exadata Database Service on Cloud@Customer
Zero Trust Packet Routing (ZPR) Oracle Cloud Infrastructure Zero Trust Packet Routing (ZPR) protects sensitive data from unauthorized access through intent-based security policies you write for resources, such as an AVMC to which you assign security attributes.	APPLIES TO:  Oracle Public Cloud only You can apply ZPR security attributes to an AVMC while provisioning it or later.	Create an Autonomous Exadata VM Cluster Getting Started with Zero Trust Packet Routing
License Type You can choose to use existing Oracle Database software licenses or subscribe to Oracle Database software licenses.	There are two options for the license type: • Bring your own license: Select this option if your organization already owns Oracle Database software licenses that you want to use on Autonomous AI Database. • License included: Select this option to subscribe to Oracle Autonomous AI Database software license	
Time Zone You can set the time-zone of each Autonomous Exadata VM Cluster.	The default time zone for the Autonomous Exadata VM Cluster is UTC, but you can specify a different time zone.	

Autonomous AI Database Feature	Notes	Further Reference
Listener You can choose a Single Client Access Name (SCAN) listener port for TLS and non-TLS, and choose between one-way TLS and mTLS for database TLS certificates.	The SCAN listener ports can not be changed after provisioning the AVMC resource. As ORDS certificates are one-way TLS certificates, choosing between one-way TLS and mutual TLS (mTLS) is applicable only to database TLS certificates.	
Encryption Keys using AWS KMS	In Autonomous AI Database on Oracle Database@AWS, you can enable AWS Key Management Service (AWS KMS) to manage Master Encryption Keys.	Enable or Disable AWS KMS
Manage certificates You can upload your CA-signed certificate to your Autonomous VM Cluster and use it for secure connections.	By default, Oracle managed self-signed certificates are automatically generated while provisioning an Autonomous Exadata VM Cluster (AVMC) and apply to all the databases created in that AVMC. You also have the option to use your own CA-signed server-side certificate with Autonomous AI Database. You can rotate both types of certificates via the console or API.	Certificate Management

Autonomous Exadata VM Cluster Management Operations

You can perform the following management operations on an Autonomous Exadata VM Cluster.

Operation	Task Instructions
Create an Autonomous Exadata VM Cluster	For Oracle Public Cloud deployments, see Create an Autonomous Exadata VM Cluster For Multicloud deployments, see Create an Autonomous VM Cluster (AVMC)
View a List of Autonomous Exadata VM Clusters	View a List of Autonomous Exadata VM Clusters
View Details of an Autonomous Exadata VM Cluster	View Details of an Autonomous Exadata VM Cluster
Change the License Type on an Autonomous Exadata VM Cluster	Change the License Type on an Autonomous VM Cluster
Configure Zero Trust Packet Routing (ZPR) for an Autonomous Exadata VM Cluster	Configure Zero Trust Packet Routing (ZPR) for an AVMC
Move an Autonomous Exadata VM Cluster to Another Compartment	APPLIES TO:  Oracle Public Cloud APPLIES TO:  Exadata Cloud@Customer Move an Autonomous Exadata VM Cluster to Another Compartment
Enable AWS Key Management Service	For Multicloud deployments, see Enable or Disable AWS KMS

Operation	Task Instructions
Manage Security Certificates for an Autonomous Exadata VM Cluster Resource	Manage Security Certificates for an Autonomous Exadata VM Cluster Resource
View Resource Usage for an Autonomous Exadata VM Cluster	View Resource Usage for an Autonomous Exadata VM Cluster
Manage Autonomous Exadata VM Cluster Resources	Manage Autonomous Exadata VM Cluster Resources
Change the Maintenance Schedule of an Autonomous Exadata VM Cluster	Change the Maintenance Schedule of an Autonomous Exadata VM Cluster
View and Manage Scheduled Maintenance of an Autonomous Exadata VM Cluster	View and Manage Scheduled Maintenance of an Autonomous Exadata VM Cluster
View Past Maintenance of an Autonomous Exadata VM Cluster	View Past Maintenance of an Autonomous Exadata VM Cluster
Terminate an Autonomous Exadata VM Cluster	Terminate an Autonomous Exadata VM Cluster

Note

For AWS Multicloud deployments, you can also terminate the Autonomous Exadata VM Cluster from the AWS console. See [Delete Autonomous VM Cluster in AWS region](#).

The above listed operations can also be achieved using API. See [API to Manage Autonomous Exadata VM Clusters](#) for further reference.

About Autonomous Container Database

Autonomous Container Database (ACD) is one of the four components of the four-level database architecture model, which is the foundation for an Autonomous AI Database on Dedicated Exadata Infrastructure. ACDs are provisioned inside an Autonomous Exadata VM Cluster (AVMC) and serve as containers for one or more Autonomous AI Databases.

You can create multiple ACD resources in a single AVMC resource, but you must create at least one before you can create any Autonomous AI Databases. To gain a comprehensive understanding of the four-layer architecture used with the an Autonomous AI Database on Dedicated Exadata Infrastructure and to understand the positioning of ACD within this architecture, refer to [Components of Autonomous AI Database on Dedicated Exadata Infrastructure](#).

ACDs provide the benefit of operating in isolation from each other, allowing you to separate Autonomous AI Databases by their intended uses. For example, you can create different ACDs for purposes like production and testing, or even have multiple ACDs using different database versions.

Although Fleet Administrators create, monitor, and manage ACDs, Application DBAs primarily use them to create Autonomous AI Databases. See [User Roles Associated with Autonomous AI Database on Dedicated Exadata Infrastructure](#) to learn more.

Autonomous Container Database Requirements

IAM Policy Requirements

You must have an Oracle Cloud Infrastructure account with privileges granted through required IAM Policies. The required policies depend on the operation you are performing. For a list of IAM policies pertaining to Autonomous Container Databases, see [Policies to Manage Autonomous Container Databases](#).

Minimum Resource Requirements

To create one Autonomous Container Database, you need at least:

- 8 ECPUs or 2 OCPUs per node
- 50GB local storage per node

Oracle Database Software Version Requirements

- To provision an ACD with Oracle Database software version as 26ai, you must choose an ECPU-based Autonomous Exadata VM Cluster (AVMC) that was created with the *DatabaseVersion* tag set to 26ai.
- Similarly, to provision an ACD with Oracle Database software version as 19c, you must choose an Autonomous Exadata VM Cluster (AVMC) whose *DatabaseVersion* tag is not set to 26ai.
- You cannot provision both 19c and 26ai ACDs inside the same AVMC.

Note

Multicloud deployments do not require a special tag and will automatically support both 19c and 26ai databases.

Database Features Managed from Autonomous Container Database

The following features of Autonomous AI Database can be defined and managed at the Autonomous Container Database (ACD) level.

Autonomous AI Database Feature	Notes	Further Reference
Oracle Database software version You can set the container database software version while provisioning an ACD.	You can choose the Oracle Database software version either from a base image version or an Autonomous AI Database software image that was created from another ACD. While selecting version from the base image, you can choose either the latest Oracle Database software version or its immediate predecessor. For example: Suppose the latest Oracle Database version supported by Autonomous AI Database is 19.26.0.1.0. Then, the Select base image drop-down lists 19.26.0.1.0 and 19.25.0.1.0. for you to choose. Autonomous Container Databases (ACD) with the 26ai database software version can only be provisioned on ECPU-based Autonomous Exadata VM Clusters (AVMC) created with the appropriate Tags. See 26ai Database Software Version Tag Requirements for details. Multicloud deployments do not require a special tag and will automatically support both 19c and 26ai databases.	-
Autonomous Data Guard Configuring Autonomous Data Guard enables you to keep your critical production databases available to mission critical applications despite failures.	You can enable Autonomous Data Guard from the Details page of an ACD and create up to two standby ACDs. The primary and secondary ACDs can also be deployed in different regions (cross-region). In a cross-region Autonomous Data Guard setup using customer-managed keys or KMS, depending on the number of Autonomous AI Databases in the primary ACD, new key versions will automatically be generated for the standby databases in the cross-region Vault.	Protect Critical Databases from Failures and Disasters Using Autonomous Data Guard

Autonomous AI Database Feature	Notes	Further Reference
Maintenance Schedule In general, Oracle schedules and performs entire fleet maintenance spread throughout each quarter and monthly infrastructure security fixes for vulnerabilities with CVSS scores greater than or equal to 7. You can let Oracle handle maintenance scheduling, or you can set a specific maintenance window when Oracle can begin maintenance operations.	You can choose between Rolling or Non-rolling maintenance methods for an ACD. If you choose non-rolling maintenance method in an Autonomous Data Guard configuration, there will be a downtime for the ACD and all associated Autonomous AI Databases until the patching completes. Optionally, you can also select Enable time-zone update. Time-zone files can only be updated using the non-rolling configuration method. You can define or modify the maintenance schedule settings for an ACD to be managed by Oracle or you can set a custom maintenance schedule. While customizing the maintenance schedule of an ACD, you can choose to skip patching for a quarter. However, you cannot skip patching for two consecutive quarters. When you choose to skip patching for a quarter, you need to select at least one month from that quarter. This acts as a fallback in case maintenance did not occur in the previous un-skipped quarter. In this scenario, Oracle will automatically perform the maintenance in the selected month, even if skipping is chosen for that quarter. You can view the number of one-off patches available for an ACD on its Details page. Clicking the Copy link next to it copies all those one-off patch numbers. When you reschedule an ACD maintenance event that is already scheduled, Oracle may place it in a queue if its Exadata Infrastructure resource or Autonomous Exadata VM Cluster resource is: • Already undergoing a maintenance update, or • Scheduled for a maintenance activity simultaneously as the ACD. You can schedule an on-demand maintenance to update RU (Release Update) along with the time-zone file or just the time-	Service Maintenance Schedule a Quarterly Maintenance Update

Autonomous AI Database Feature	Notes	Further Reference
	zone file for an ACD. You can also choose to update using an existing custom database software image. You may experience downtime for your ACD and the associated Autonomous AI Databases depending on the configuration of your ACD's maintenance schedule.	

Autonomous AI Database Feature	Notes	Further Reference
Backup and Restore To support high availability, Autonomous AI Database automatically backs up your database for you. The retention period for backups is up to 95 days depending on the backup retention policy/period chosen for the ACD. You can restore and recover your database to any point-in-time in this retention period.	Once enabled, automatic backups can not be disabled for an ACD. With automatic backups enabled, you have options to additionally take cross-region backups, long-term backups, and manual backups. You can define the backup retention policy/period while provisioning an ACD or modify it later from its details page on the Oracle Cloud Infrastructure console console. - You can retain the backups of a terminated ACD for either up to 72 hours or until the end of its backup retention period. - You can enable the backup retention lock to prevent any unwanted deletion of the ACD and Autonomous AI Database backups before their retention period. - You can enable the retention lock either while provisioning the ACD or later. - Once enabled, you can disable the retention lock within 14 days, beyond which the setting becomes permanent. - Once the retention lock is enabled, you can only increase the value of the retention period, and it cannot be reduced. - All long-term backups will also be locked and cannot be deleted when the retention lock is enabled. The long-term backups are automatically deleted after they expire. However, in the case of Exadata Cloud@Customer deployments, if the backup destination is NFS, then none of the backups are automatically deleted when an ACD is terminated. You need to manually delete all the automatic, manual, and long-term backups after the ACD is terminated and the expiration date has passed. See Backup Retention Policy for more details on the backup retention policy values for	Backup and Restore Autonomous AI Databases.

Autonomous AI Database Feature	Notes	Further Reference
	different Autonomous AI Database deployments.	
Backup Destination A backup destination defines the properties that are required to connect to a backup location, and each backup destination must be accessible in your data center from the VM cluster nodes.	The ability to choose a backup destination while provisioning an ACD, and the supported backup destinations vary depending on the deployment platform. See Backup destination for more details on the different backup destination types.	To learn more about configuring NFS backup destination for Cloud@Customer, see Prerequisites for Backup Destinations for Exadata Cloud@Customer. Refer to Edit Autonomous Container Database Backup Settings for instructions on changing the backup destination type after provisioning the ACD.
NFS Space Utilization	APPLIES TO:  Exadata Cloud@Customer only If the current backup destination type is NFS, then the current NFS space utilization is displayed as a percentage along with a status icon.	See View NFS Space Utilization for more details.
Resource Management Attributes Resource management attributes affect how resources are managed to either consolidate more databases or have the highest database availability.	Optionally, while provisioning an ACD, you can define a suitable value for the following resource management attributes to suit your needs: • Database split threshold (CPU): The CPU value beyond which an Autonomous AI Database will be opened across multiple nodes. The default value of this attribute is 16 for OCPUs and 64 for ECPUs. • Node failover reservation (%): Determines the percentage of CPUs reserved across nodes to support node failover. Allowed values are 0%, 25%, and 50%, with 50% being the default option. • Distribution affinity: Determines whether an Autonomous AI Database must be opened across a minimum or maximum of nodes. By default, Minimum nodes is selected.	For more information on how these ACD attributes affect the performance of your databases, see CPU Billing Details.
Shared Server Connections The shared server architecture enables a database server to allow many client processes to share very few server processes, so the number of users that can be supported is increased.	While provisioning an ACD, you can optionally enable shared server connections. You cannot disable shared server architecture after provisioning the ACD.	Special-Purpose Connection Features.

Autonomous AI Database Feature	Notes	Further Reference
Encryption Key By default, Autonomous AI Database creates and manages all the master encryption keys used to protect your data, storing them in a secure PKCS 12 keystore on the same Exadata systems where the databases reside. If your company security policies require, Autonomous AI Database can instead use keys you create and manage.	While provisioning an ACD, you can optionally configure the ACD to use customer-managed encryption keys instead of Oracle-managed encryption keys. You can choose between the following options when using customer-managed encryption keys: • OCI Vault Service: With this option you select a Vault and a Master Encryption Key. This option is only available on Oracle Public Cloud. • Oracle Key Vault: You select a Key Store with this option and enter an OKV Endpoint group name. • AWS KMS Service: In Autonomous AI Database on Oracle Database@AWS, you can use AWS Key Management Service (AWS KMS) to manage Master Encryption Keys. You can use customer-managed encryption keys with Autonomous Data Guard-enabled ACDs with the primary and standby databases located in different availability domains within the same region.	About Master Encryption Keys Use Bring Your Own Keys (BYOK) in Vault Service Use Customer-Managed Keys in Oracle Key Vault Rotate the Encryption Key of an Autonomous Container Database.
Contact Email You can provide contact emails where you can receive operational notifications, announcements, and unplanned maintenance notifications regarding your Autonomous Container Database.	Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.	
Full Stack Disaster Recovery Full Stack DR is an Oracle Cloud Infrastructure (OCI) disaster recovery orchestration and management service that provides comprehensive disaster recovery capabilities for all layers of an application stack, including infrastructure, middleware, database, and application.	You can enable the OCI Full Stack Disaster Recovery and use it to perform switchover/failover operations or optionally perform database only Autonomous AI Database switchover/failover operations.	Use OCI Full Stack Disaster Recovery on Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database Feature	Notes	Further Reference
Cross Region Backup Copy You can select a secondary region for a copy of your backups. In case of a region failure, you can clone the ACD's remote backup to restore the data. The backup of an ACD can only be cloned to an Autonomous Exadata VM Cluster (AVMC) different from its source AVMC.	APPLIES TO:  Oracle Public Cloud only Cross region backup copy can be enabled while provisioning an ACD or from the details page of an existing ACD. You cannot enable cross region backup if Autonomous Data Guard is enabled.	About Cloning Autonomous Container Database on Dedicated Exadata Infrastructure Clone an Autonomous Container Database

Autonomous Container Database Management Operations

You can perform the following management operations on an Autonomous Container Database.

Operation	Task Instructions
Create an Autonomous Container Database	Create an Autonomous Container Database
Change the backup retention policy of an Autonomous Container Database	Edit Autonomous Container Database Backup Settings
Create an Autonomous AI Database software image	Create an Autonomous AI Database Software Image
Edit the maintenance preferences of an Autonomous Container Database	Update Autonomous Container Database Maintenance Preferences
Manage Autonomous Data Guard Configuration	Manage Autonomous Data Guard Configuration
Move an Autonomous Container Database to another compartment	Move an Autonomous Container Database to a Different Compartment
Rotate an Autonomous Container Database encryption key	Rotate the Encryption Key of an Autonomous Container Database
Restart an Autonomous Container Database	Restart an Autonomous Container Database
Terminate an Autonomous Container Database	Terminate an Autonomous Container Database
Manage Customer Contacts for an Autonomous Container Database	Manage Customer contacts for an Autonomous Container Database
View a list of Autonomous Container Databases	View a list of Autonomous Container Databases
View details of an Autonomous Container Database	View details of an Autonomous Container Database
View NFS space utilization	APPLIES TO:  Exadata Cloud@Customer only View NFS Space Utilization

The above listed operations can also be achieved using API. See API to Manage Autonomous Container Databases for further reference.

Monitoring Autonomous Container Database

You can use dynamic performance views to monitor your Autonomous Container Database (ACD) in the following ways:

- View real-time metrics on different wait events and wait classes.
- View historical snapshots of wait class metrics.

- View real-time, historical, and summary performance metrics data.
- View resource limits and current usage.

For detailed information, see [Dynamic Performance Views](#).

About Autonomous AI Database

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure is a highly automated, fully managed database environment running in Oracle Cloud Infrastructure (OCI) with committed hardware and software resources. These isolated resources enable organizations to meet stringent security, availability, and performance requirements while reducing cost and complexity. Autonomous AI Databases are provisioned inside an Autonomous Container Database (ACD) and are user databases. You can create many Autonomous AI Databases in a single Autonomous Container Database resource. An Autonomous AI Database will inherit the retention lock settings and associated backup rules from its parent ACD.

To get a holistic idea of the four layer architecture used with Autonomous AI Database on Dedicated Exadata Infrastructure and understand Autonomous AI Database's positioning in this architecture, refer to [Components of Autonomous AI Database on Dedicated Exadata Infrastructure](#).

Application DBAs create, monitor and manage Autonomous AI Databases. Additionally, they create and manage Oracle Database users within these databases, and provide others the information necessary access the database. You can connect to your Autonomous AI Database and start developing database applications after your fleet administrator sets up infrastructure resources and an Application DBA provisions an Autonomous AI Database with database users. A database user can also connect to Autonomous AI Database using various tools such as SQL*Plus, SQLcl, Database Actions, or Oracle APEX.

Autonomous AI Database Requirements

Resource Requirements

To provision an Autonomous AI Database, you need an Autonomous Container Database with or without Autonomous Data Guard, depending on disaster recovery requirements. See [Create an Autonomous Container Database](#) for details.

Required IAM Policies

You must have an Oracle Cloud Infrastructure account with privileges granted through required IAM Policies. The required policies depend on the operation you are performing. For a list of IAM policies pertaining to Autonomous AI Databases, see [Policies to Manage Autonomous AI Databases](#).

Database Features Managed from Autonomous AI Database

The following features can be defined and managed at the Autonomous AI Database level.

Feature	Notes	Further Reference
Workload Type You can configure your database to be one of either Autonomous AI Lakehouse or Autonomous AI Transaction Processing workload types.	Oracle Autonomous AI Lakehouse is a cloud data warehouse service that eliminates virtually all the complexities of operating a data warehouse, securing data, and developing data-driven applications. Oracle Autonomous AI Transaction Processing is a fully automated database service optimized to run transactional, analytical, and batch workloads concurrently.	About Autonomous AI Database on Dedicated Exadata Infrastructure
View Connection Details You can download client credentials and view the TNS names and connection strings of an Autonomous AI Database from the Oracle Cloud Infrastructure (OCI) console.	Oracle client credentials (wallet files) are downloaded from Autonomous AI Database by a service administrator. If you are not an Autonomous AI Database administrator, your administrator should provide you with the client credentials. For cross-region standby Autonomous AI Databases, you can download region-specific connection strings.	About Connecting to a Dedicated Autonomous AI Database
Autonomous AI Database for Developers You can create an Autonomous AI Database for Developers instance. Autonomous AI Database for Developers is a free tier offering designed for database development and functional testing activities.	Autonomous AI Database for Developers comes fixed at 4 ECPUs and 32GB storage, and do not support manual or auto-scaling. As developer database instances can only be created on an ECPU-based ACDs without Autonomous Data Guard, the Free instance toggle button is disabled for ACDs with OCPU, Autonomous Data Guard, or both.	Autonomous AI Database for Developers
CPU Count You can select the number of CPUs for your database from the list of provisionable CPUs.	The CPU type, that is, ECPU or OCPU is determined by the parent Autonomous Exadata VM Cluster's compute type. This value defaults to 2 ECPUs or 1 OCPU depending on the CPU type.	Compute Management in Autonomous AI Database
CPU Auto Scaling CPU auto scaling permits Autonomous AI Database to automatically use up to three times as many CPUs as specified by CPU Count as the workload on the database increases.	You can enable CPU Auto Scaling while provisioning an Autonomous AI Database or after one has already been provisioned.	Enable or Disable Auto Scaling of an Autonomous AI Database

Feature	Notes	Further Reference
Storage You can specify the storage to allocate to your database in terabytes (TB) while provisioning an Autonomous AI Database.	The minimum value is 32 GB. The default values are 1024 GB for Autonomous AI Lakehouse and 32 GB for the Autonomous AI Transaction Processing workloads.	
Elastic Pool Elastic pools help you improve operating efficiency and reduce costs by bringing all of your databases to the cloud. This also supports consolidating resources and simplifying administration and operations by using Autonomous AI Database.	With Compute auto scaling disabled, you can choose to create an elastic pool as a pool leader or join an existing elastic pool as a pool member using: - The elastic pool options under the Configure the database section, while: - Provisioning a new Autonomous AI Database - Cloning an Autonomous AI Database or its backup - The elastic pool options on the Manage resource allocation dialog from the Autonomous AI Database Details page.	Only Autonomous AI Transaction Processing databases without Autonomous Data Guard that use ECPU compute model can be used to create an elastic pool.
Database Authentication You set the database username and password while provisioning a database.	The user name and password is defined while creating the database will be the ADMIN username and password. An Application DBA can connect to this database using the ADMIN username and password to create other Database users.	Create Database Users
Access Control You can configure network access by creating an access control list (ACL). An ACL provides additional protection to your Autonomous AI Database by allowing only the client with specific IP addresses to connect to the database. If the parent Autonomous Container Database uses Autonomous Data Guard, you can define access control for the standby database also.	Depending on the types of addresses in your list, you can choose one of the following IP notation type options: IP Address: Specify individual IP addresses. CIDR Block: Specify ranges of public IP addresses using CIDR notation.	Access Control Within Autonomous AI Database on Dedicated Exadata Infrastructure
Contact Email You can provide contact emails where you can receive operational notifications, announcements, and unplanned maintenance notifications regarding your Autonomous AI Database.	Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.	

Feature	Notes	Further Reference
Character Set Autonomous AI Database lets you choose a character set of your choice from a list of supported character sets while provisioning an Autonomous AI Database.	You choose a Character Set and National Character Set while provisioning an Autonomous AI Database. The list of supported character sets currently includes all database character sets supported on ASCII-based platforms.	Character Set Selection for Autonomous AI Database
Database In-Memory You can enable Database In-memory for your Autonomous AI Database by allocating a percentage of its System Global Area (SGA) to the In-Memory column store (IM column store) either while provisioning the database or later. The In-Memory Column Store (IM column store) is the key feature of Database In-Memory. The IM column store maintains copies of tables, partitions, and individual columns in a special compressed columnar format optimized for rapid scans.	You can enable or disable Database In-Memory for an existing Autonomous AI Database from its Details page on the Oracle Cloud Infrastructure (OCI) console. You can also enable Database In-Memory on databases cloned from a database instance or backup, irrespective of whether the clone source has Database In-memory enabled.	Database In-Memory
Database Cloning You can clone an Autonomous AI Database, creating a point-in-time copy of it or its backup set. You can use the cloning feature to quickly set up an Autonomous AI Database with historical data for purposes such as testing, development, or analytics.	Autonomous AI Database supports the following clone types: • Full Clone: A full clone creates a new database that includes the metadata and data from the source database. • Metadata Clone: This clone type creates a new database that includes all source database schema metadata, but not the source database data.	About Cloning Autonomous AI Database on Dedicated Exadata Infrastructure

Feature	Notes	Further Reference
Ops Insights Ops Insights is a cloud-native service that provides 360-degree insight into the resource utilization and capacity of databases and hosts. You can easily analyze CPU and storage resources, forecast capacity issues, and proactively identify SQL performance issues across your database fleet.	By default, Ops Insights is disabled for an Autonomous AI Database, and you must enable it from the Oracle Cloud Infrastructure console. With Ops Insights, you can: - Analyze resource usage of databases/hosts across the enterprise. - Forecast future demand for resources based on historical trends. - Compare SQL Performance across databases and identify common patterns. - Identify SQL performance trends across enterprise-wide databases. - Analyze AWR statistics for database performance, diagnostics, and tuning across a fleet of databases.	Use Operations Insights on Autonomous AI Database on Dedicated Exadata Infrastructure Get Started with Ops Insights
Autonomous AI Database Metrics You can monitor the health, capacity, and performance of your Autonomous AI Databases with metrics, alarms, and notifications. The Autonomous AI Database metrics help you measure useful quantitative data, such as CPU and storage utilization, the number of successful and failed database logon and connection attempts, database operations, SQL queries, and transactions, and so on. You can use metrics data to diagnose and troubleshoot problems with your Autonomous AI Database resources.	You can use Oracle Cloud Infrastructure console or Monitoring APIs to view metrics.	Monitor Databases with Autonomous AI Database Metrics

Feature	Notes	Further Reference
Data Safe Oracle Data Safe helps you understand the sensitivity of your data, evaluate risks to data, mask sensitive data, implement and monitor security controls, assess user security, monitor user activity, and address data security compliance requirements in your databases.	Oracle Data Safe provides the following set of features in a single, easy-to-use management console: • Security Assessment helps you assess the security of your database configuration. • User Assessment helps you assess the security of your database users and identify high risk users. • Data Discovery helps you find sensitive data in your database. Data Masking provides a way for you to mask sensitive data so that the data is safe for non-production purposes. • Activity Auditing lets you audit user activity on your database so you can monitor database usage and be alerted of unusual database activities. Before you can register your database with Data Safe, Data Safe must be configured to access databases in your dedicated infrastructure configuration.	Oracle Data Safe Overview

Autonomous AI Database Tools

The following tools are available to Autonomous AI Database users:

Tool	Notes	Further Reference
Database Actions Oracle Database Actions is a browser-based application that provides development tools, data tools, administration, and monitoring features for Autonomous AI Database. Using Database Actions, you can load data and run SQL statements, queries, and scripts in a worksheet, export data, create Data Modeler diagrams, and enable database administrators to monitor the database.	You can connect to Autonomous AI Database using Database Actions without downloading or installing additional software on your system. Oracle Database Actions runs in Oracle REST Data Services and access to it is provided through schema-based authentication. To use Oracle Database Actions, you must sign in as a database user whose schema has been enabled for Database Actions.	About Oracle Database Actions Connect to Autonomous AI Database with Database Actions

Tool	Notes	Further Reference
Oracle APEX (Oracle Application Express) Oracle APEX provides you with an easy-to-use browser-based environment to load data, manage database objects, develop REST interfaces, and build applications.	Configuration, patching, monitoring, and upgrading all Oracle Application Express components are fully managed by Oracle. There are no limits on the number of developers or end-users for your Oracle APEX applications. Autonomous AI Database can instantly scale compute and storage online as needed, based upon your workload. You can deploy the Oracle APEX applications developed on-premise to Oracle APEX on Autonomous AI Database, or vice-versa easily.	Build an APEX Application
SQL*Plus SQL*Plus is a command-line interface used to enter SQL commands. You can connect to Autonomous AI Database using SQL*Plus to define the database's tables, views, triggers, types, sequences and so on.	To be able to connect SQL*Plus to an Autonomous AI Database, the system running SQL*Plus must have network access to the Autonomous AI Database.	Connect with SQL*Plus
SQLcl (Oracle SQL Developer Command Line) SQLcl 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.	To be able to connect SQLcl to an Autonomous AI Database, the system running SQLcl must have network access to the Autonomous AI Database.	Connect with Oracle SQLcl
SQL Developer Oracle SQL Developer is a free integrated development environment that simplifies the development and management of Oracle Database in both traditional and cloud deployments. SQL Developer offers complete end-to-end development of your PL/SQL applications, a worksheet for running queries and scripts, a DBA console for managing the database, a reports interface, a complete data modeling solution, and a migration platform for moving your 3rd party databases to Oracle.	To be able to connect SQL Developer to an Autonomous AI Database, the system running SQL Developer must have network access to the Autonomous AI Database.	Connect with Oracle SQL Developer

Tool	Notes	Further Reference
Oracle REST Data Services Oracle REST Data Services (ORDS) makes it easy to develop REST interfaces for relational data in an Autonomous AI Database. ORDS is a mid-tier Java application that maps HTTP(S) verbs, such as GET, POST, PUT, DELETE, and so on, to database transactions, and returns any results as JSON data.	The Oracle REST Data Services (ORDS) application in Autonomous AI Database is preconfigured and fully managed. ORDS connects to the database using the low predefined database service with a fixed maximum number of connections (the number of connections for ORDS does not change based on the number of CPUs). It is not possible to change the default ORDS configuration.	Developing RESTful Services in Autonomous AI Database
Simple Oracle Document Access (SODA) Oracle provides a family of Simple Oracle Document Access (SODA) APIs for access to JSON data stored in the database. SODA is designed for schemaless application development without knowledge of relational database features or languages such as SQL and PL/SQL. It lets you create and store collections of documents in Oracle Database, retrieve them, and query them, without needing to know how the documents are stored in the database.	You can download the SODA drivers from the Details page of an Autonomous AI Database. There are available implementations of SODA are: • SODA for REST • SODA for Java: • SODA for PL/SQL • SODA for C • SODA for Node.js • SODA for Python	Overview of SODA
Oracle Database API for MongoDB Oracle Database API for MongoDB translates the MongoDB wire protocol into SQL statements that are executed by Oracle Database. It lets developers who have MongoDB skill sets write JSON document-store applications for Oracle Database that use drivers and tools that understand the MongoDB protocol.	To use the MongoDB API with an Autonomous AI Database, you must install and configure customer managed Oracle REST Data Services (ORDS) separately, and the version of ORDS must be 22.3 or later.	Overview of Oracle Database API for MongoDB

Autonomous AI Database Management Operations

You can perform the following management operations on an Autonomous AI Database.

Operation	Task Instructions
Create an Autonomous AI Database	Create an Autonomous AI Database
Create an Elastic Pool	Create an Elastic Pool
Backup Your Autonomous AI Database Manually	Backup an Autonomous AI Database Manually
Clone an Autonomous AI Database	Clone an Autonomous AI Database
Create a Long-Term Backup	Create a Long-Term Backup

Operation	Task Instructions
Enable or Disable Auto Scaling of an Autonomous AI Database	Enable or Disable Auto Scaling of an Autonomous AI Database
Enable or Disable Database In-Memory	Enable or Disable Database In-Memory
Enable Ops Insights for an Autonomous AI Database	Enable Ops Insights for an Autonomous AI Database
Join or Leave an Elastic Pool	Join or Leave an Elastic Pool
Manage an Elastic Pool as a Pool Leader	Manage an Elastic Pool as a Pool Leader
View Details of an Autonomous AI Database	View Details of an Autonomous AI Database
Manage Customer Contacts for an Autonomous AI Database	Manage Customer Contacts for an Autonomous AI Database
Manage CPU or Storage Resources of an Autonomous AI Database	Manage CPU or Storage Resources of an Autonomous AI Database
Manage Long-Term Backups	Manage Long-Term Backups
Manage Primary and Standby Databases in Autonomous Data Guard Configuration	Manage Primary and Standby Databases in an Autonomous Data Guard Configuration
Move an Autonomous AI Database to a Different Compartment	Move an Autonomous AI Database to a Different Compartment
Restore and Recover Your Autonomous AI Database	Restore and Recover Autonomous AI Database
Start, Stop, and Restart an Autonomous AI Database	Start, Stop, and Restart an Autonomous AI Database
View Autonomous AI Database Metrics for a Database	View Autonomous AI Database Metrics for a Database
View Autonomous AI Database Metrics for Databases in a Compartment	View Autonomous AI Database Metrics for Databases in a Compartment
Download Client Credentials	Download Client Credentials
View Connection Strings for an Autonomous AI Database	View Connection Strings for an Autonomous AI Database
Set Access Control List for an Autonomous AI Database	Set Access Control List for an Autonomous AI Database
Register or Deregister a Dedicated Database with Data Safe	Register or Deregister a Dedicated Database with Data Safe
Terminate an Autonomous AI Database	Terminate an Autonomous AI Database
Undelete an Autonomous AI Database	Undelete an Autonomous AI Database

The above listed operations can also be achieved using API. See [API to Manage Autonomous AI Databases](#) for further reference.

Autonomous AI Database on Dedicated Exadata Infrastructure in AWS Region

Using Autonomous AI Database on Oracle Database@AWS, you can provision Dedicated Exadata Infrastructure and Autonomous VM Clusters within Amazon Web Service (AWS) that will host Multicloud deployments of Autonomous AI Database.

This simplifies the migration and deployment of enterprise workloads to the cloud while improving availability, scalability, performance, and security. Multicloud provides you with a unified experience between Oracle Cloud Infrastructure (OCI) and AWS, offering simplified database administration, billing, and unified customer support. You will be able to seamlessly

and securely connect and analyze data in the Autonomous AI Database using applications that are running on AWS to get faster transactions and deeper insights. You get the most mission-critical Oracle Database capabilities in a fully managed service while leveraging a seamless integration with AWS services such as CloudWatch with the option to use OCI Object Storage, or Amazon Simple Storage (S3), or Autonomous Recovery Service (recommended option) to store your service automated database backups.

Prerequisites

You need to obtain the following before you create various Autonomous AI Database resources with Multicloud.

- An AWS account
- An OCI account
- Required OCI IAM Permissions and AWS IAM Permissions.

See [Prerequisites for Oracle Database@AWS](#) for more details.

Before you begin provisioning Oracle Database@AWS resources, you need to check if your existing OCI limits will let you deploy the number of Oracle Database@AWS resources you need. Else, you need to request an increase in the service limits. See [Service Limits for Oracle Database@AWS](#) for more details.

Creating Autonomous AI Database Resources in AWS Region

✓ Tip

For a "try it out" alternative that demonstrates the tasks listed below, run the labs in the workshop Oracle Database@AWS- Autonomous AI Database on Dedicated Infrastructure.

With Autonomous AI Database on Oracle Database@AWS, you can perform the following tasks from the AWS console:

- [Create an ODB Network](#)
- [Create an Exadata Infrastructure \(EI\)](#)
- [Create an Autonomous VM Cluster \(AVMC\)](#)

i Note

Multicloud AVMCs deployed in AWS regions support Oracle Database versions 19c and 23ai without any additional tag requirements.

If you want to use AWS Key Management Service (AWS KMS) to manage Master Encryption Keys, then you need to perform the following tasks:

- [Configure OCI Identity Domain](#)
- [Create or Use an Existing ODB Network](#)
- [Configure Identity Provider](#)
- [Associate an IAM Role to an AVMC](#)

- [Create a Key in AWS KMS](#)

Once an EI and AVMC are provisioned using the AWS console, you can create the Autonomous Container Database (ACD) and Autonomous AI Database using the OCI console. You can provision these resources in one of the following two ways:

- As an existing OCI customer with a Multicloud subscription, you can log in to the OCI console and browse the Multicloud resources (EI and AVMC) in the AWS-specific compartment. You can then create resources, such as ACD and Autonomous AI Database, using Exadata Infrastructure and an Autonomous VM Cluster provisioned in the AWS-specific compartment.
- As a customer with a Multicloud subscription, you can navigate to the Oracle Database@AWS landing page and view the details of the VM cluster in the AWS region. You can also click **Manage in OCI** to redirect to the OCI console and view the AVMC resource details page. See Autonomous AI Database in the OCI Console for details.

While creating an ACD, you have the option to use AWS KMS to manage Master Encryption Keys. However to do so, you need to enable AWS key management for the parent AVMC. You can also optionally disable the AWS KMS key management configuration from the AVMC details page before creating an ACD. See Enable or Disable AWS KMS for details.

See Create an Autonomous Container Database and Create an Autonomous AI Database on Dedicated Exadata Infrastructure for instructions to create ACD and Autonomous AI Database from the OCI console.

Managing Oracle Database@AWS Resources

You can manage EI and AVMC from both AWS and OCI consoles. However, you can manage ACD and Autonomous AI Database using the OCI console only.

Management Functions from the AWS Console

After provisioning an Autonomous AI Database on Oracle Database@AWS, you can use the AWS console for a limited set of management functions listed below.

- [Modify EI in the AWS region](#)
- [Delete EI in the AWS region](#)
- [View a list of AVMCs created in the AWS region](#)
- [View details of a specific AVMC in the AWS region](#)
- [Modify an AVMC in the AWS region](#)
- [Delete an AVMC created in the AWS region](#)

Management Functions from the OCI Console

You can also manage Autonomous AI Database on Oracle Database@AWS resources (EI, AVMC, ACD, and Autonomous AI Database) from the OCI console.

You can navigate to the OCI console in either of the following two ways:

- As an existing OCI customer with a Multicloud subscription, you can log in directly to the OCI console.
- As a customer with a Multicloud subscription, you can navigate to the Autonomous AI Database on Oracle Database@AWS landing page and view the details of the VM cluster in the AWS region. Clicking **Manage in OCI** redirects you to the AVMC resource details page in the OCI console. See Autonomous AI Database in the OCI Console for details.

You can perform the following actions from the OCI console.

Table 1-1 Management Functions from the OCI Console

Feature	Additional Reference
Change the Maintenance Schedule of an Exadata Infrastructure Resource	Change the Maintenance Schedule of an Exadata Infrastructure Resource
Manage Contacts for an Exadata Infrastructure	Manage Customer contacts for an Exadata Infrastructure
View Resource Usage for an Autonomous Exadata VM Cluster	View Resource Usage for an Autonomous Exadata VM Cluster
Manage Autonomous Exadata VM Cluster Resources	Manage Autonomous Exadata VM Cluster Resources
Change the Maintenance Schedule of an Autonomous Exadata VM Cluster	Change the Maintenance Schedule of an Autonomous Exadata VM Cluster
Edit Autonomous Container Database Backup Setting	Edit Autonomous Container Database Backup Setting
Manage Autonomous Data Guard Configuration	Manage Autonomous Data Guard Configuration
Manage Customer Contacts for an Autonomous Container Database	Manage Customer Contacts for an Autonomous Container Database
Restart an Autonomous Container Database	Restart an Autonomous Container Database
Schedule a Quarterly Maintenance Update	Schedule a Quarterly Maintenance Update
Terminate an Autonomous Container Database	Terminate an Autonomous Container Database
View a List of Autonomous Container Databases	View a List of Autonomous Container Databases
View Details of an Autonomous Container Database	View Details of an Autonomous Container Database
View Resource Usage for an Autonomous Container Database	View Resource Usage for an Autonomous Container Database
View and Manage Scheduled Maintenance of an Autonomous Container Database	View and Manage Scheduled Maintenance of an Autonomous Container Database
Update Autonomous Container Database Maintenance Preferences	Update Autonomous Container Database Maintenance Preferences
Move an Autonomous Container Database to a Different Compartment	Move an Autonomous Container Database to a Different Compartment
View Details of an Autonomous AI Database	View Details of an Autonomous AI Database
Manage Customer Contacts for an Autonomous AI Database	Manage Customer Contacts for an Autonomous AI Database
Backup an Autonomous AI Database Manually	Backup an Autonomous AI Database Manually
Move an Autonomous AI Database to a Different Compartment	Move an Autonomous AI Database to a Different Compartment
Create a Long-Term Backup	Create a Long-Term Backup
Enable or Disable Auto Scaling of an Autonomous AI Database	Enable or Disable Auto Scaling of an Autonomous AI Database
Enable or Disable Database In-Memory	Enable or Disable Database In-Memory
View Autonomous AI Database Metrics for a Database	View Autonomous AI Database Metrics for a Database
Manage CPU or Storage Resources of an Autonomous AI Database	Manage CPU or Storage Resources of an Autonomous AI Database
Clone an Autonomous AI Database	Clone an Autonomous AI Database
Manage Long-Term Backups	Manage Database backups
Restore and Recover Autonomous AI Database	Restore and Recover Autonomous AI Database

Table 1-1 (Cont.) Management Functions from the OCI Console

Feature	Additional Reference
Start, Stop, and Restart an Autonomous AI Database	Start, Stop, and Restart an Autonomous AI Database
Terminate an Autonomous AI Database	Terminate an Autonomous AI Database

Observe Oracle Database@AWS

You can observe Oracle Database@AWS from the AWS console using aggregate metrics from both AWS and OCI for a unified monitoring solution. See the [Prerequisites](#) before using the following monitoring and management services in your Oracle Database@AWS.

- **CloudWatch:** A fully managed observability service that enables you to collect, analyze, and act upon operational data in real time. See [CloudWatch](#) for more information.
- **EventBridge:** Delivers a continuous flow of real-time data from applications and AWS services by forwarding data to different targets, such as **AWS Lambda** and **Amazon Simple Notification Service**. See [EventBridge](#) for more information.
- **CloudTrail:** Enables governance, compliance, operational auditing, and risk auditing of your AWS account. See [CloudTrail](#) for more information.

Unsupported Features in Oracle Database@AWS

The following features are not currently supported in Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@AWS.

- Zero Trust Packet Routing (ZPR) Security Attributes
- Move an Exadata Infrastructure created in an AWS region to another compartment
- Move an AVMC created in an AWS region to another compartment
- Cross-region Disaster Recovery based backup for S3

Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@Azure

Using Autonomous AI Database on Oracle Database@Azure, you can provision Dedicated Exadata Infrastructure inside Azure data centers that will host Multicloud deployments of Autonomous AI Database. Oracle Database@Azure offers a powerful and flexible solution for running Oracle workloads directly within Azure, giving your business-critical applications low-latency connectivity to your Oracle Databases.

Prerequisites

You need to obtain the following before you create various Autonomous AI Database resources in Azure Multicloud deployments:

- An Azure account
- An OCI account
- Required OCI IAM Permissions and Azure IAM Permissions.

See [Prerequisites for Oracle Database@Azure](#) for more details.

Before you begin provisioning Oracle Database@Azure resources, you need to check if your existing OCI limits will let you deploy the number of Oracle Database@Azure resources you need. Else, you need to request an increase in the service limits. See [Service Limits for Oracle Database@Azure](#) for more details.

Creating Autonomous AI Database Resources in Azure Region

With Autonomous AI Database on Oracle Database@Azure, you can perform the following tasks from the Azure console:

- [Create a Resource Anchor](#)
- [Create a Network Anchor](#)
- [Create Exadata Infrastructure \(EI\)](#)

Once an EI is provisioned using the Azure console, you can create the Autonomous Exadata VM Cluster (AVMC), Autonomous Container Database (ACD), and Autonomous AI Database resources using the OCI console. For instructions, see:

- [Create an Autonomous Exadata VM Cluster](#)
- [Create an Autonomous Container Database](#)
- [Create an Autonomous AI Database on Dedicated Exadata Infrastructure](#)

Managing Oracle Database@Azure Resources

You can manage EI from both the Azure and OCI consoles. However, you can manage AVMC, ACD, and Autonomous AI Database resources using the OCI console only.

Management Functions from the Azure Console

you can use the Azure console for a limited set of management functions listed below.

- [Modify Exadata Infrastructure in the Azure Region](#)
- [Delete Exadata Infrastructure in the Azure Region](#)

Management Functions from the OCI Console

You can manage Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@Azure resources (EI, AVMC, ACD, and Autonomous AI Database) from the OCI console. You can navigate to the OCI console in either of the following two ways:

- As an existing OCI customer with a Multicloud subscription, you can log in directly to the OCI console.
- As a customer with a Multicloud subscription, you can navigate to the Exadata Infrastructure on Oracle Database@Azure landing page and view the **Overview** page of the EI in the Azure region. Expanding the **Essentials** section and then clicking **Open in OCI** redirects you to the EI resource details page in the OCI console.

You can perform the following actions from the OCI console.

Table 1-2 Management Functions from the OCI Console

Feature	Additional Reference
Change the Maintenance Schedule of an Exadata Infrastructure Resource	Change the Maintenance Schedule of an Exadata Infrastructure Resource
Manage Contacts for an Exadata Infrastructure	Manage Customer contacts for an Exadata Infrastructure
View Resource Usage for an Autonomous Exadata VM Cluster	View Resource Usage for an Autonomous Exadata VM Cluster
Manage Autonomous Exadata VM Cluster Resources	Manage Autonomous Exadata VM Cluster Resources
Change the Maintenance Schedule of an Autonomous Exadata VM Cluster	Change the Maintenance Schedule of an Autonomous Exadata VM Cluster
Edit Autonomous Container Database Backup Setting	Edit Autonomous Container Database Backup Setting
Manage Autonomous Data Guard Configuration	Manage Autonomous Data Guard Configuration
Manage Customer Contacts for an Autonomous Container Database	Manage Customer Contacts for an Autonomous Container Database
Restart an Autonomous Container Database	Restart an Autonomous Container Database
Schedule a Quarterly Maintenance Update	Schedule a Quarterly Maintenance Update
Terminate an Autonomous Container Database	Terminate an Autonomous Container Database
View a List of Autonomous Container Databases	View a List of Autonomous Container Databases
View Details of an Autonomous Container Database	View Details of an Autonomous Container Database
View Resource Usage for an Autonomous Container Database	View Resource Usage for an Autonomous Container Database
View and Manage Scheduled Maintenance of an Autonomous Container Database	View and Manage Scheduled Maintenance of an Autonomous Container Database
Update Autonomous Container Database Maintenance Preferences	Update Autonomous Container Database Maintenance Preferences
Move an Autonomous Container Database to a Different Compartment	Move an Autonomous Container Database to a Different Compartment
View Details of an Autonomous AI Database	View Details of an Autonomous AI Database
Backup an Autonomous AI Database Manually	Backup an Autonomous AI Database Manually
Move an Autonomous AI Database to a Different Compartment	Move an Autonomous AI Database to a Different Compartment
Create a Long-Term Backup	Create a Long-Term Backup
Enable or Disable Auto Scaling of an Autonomous AI Database	Enable or Disable Auto Scaling of an Autonomous AI Database
Enable or Disable Database In-Memory	Enable or Disable Database In-Memory
View Autonomous AI Database Metrics for a Database	View Autonomous AI Database Metrics for a Database
Manage CPU or Storage Resources of an Autonomous AI Database	Manage CPU or Storage Resources of an Autonomous AI Database
Clone an Autonomous AI Database	Clone an Autonomous AI Database
Manage Long-Term Backups	Manage Database backups
Restore and Recover Autonomous AI Database	Restore and Recover Autonomous AI Database
Start, Stop, and Restart an Autonomous AI Database	Start, Stop, and Restart an Autonomous AI Database
Terminate an Autonomous AI Database	Terminate an Autonomous AI Database

Unsupported Features in Oracle Database@Azure

The following features are not currently supported in Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@Azure:

- Autonomous Recovery Service
- Azure Blob Storage
- Azure KMS
- Azure Monitor, including Azure Log Analytics

Autonomous AI Database for Developers

Autonomous AI Database for Developers instances are free Autonomous AI Databases that developers can use to build and test new applications.

With Autonomous AI Database for Developers instances, you can try new Autonomous AI Database features for free and apply them to ongoing or new development projects. Developer database comes with limited resources and is, therefore, not suitable for large-scale testing and production deployments. When you need more compute or storage resources, you can transition to a paid database licensing by cloning your developer database into a regular Autonomous AI Database.

Requirements

To create an Autonomous AI Database for Developers, you must have access to Oracle Exadata Database Service or Autonomous AI Database on either a Dedicated Exadata Infrastructure or Exadata Cloud@Customer. In other words, only those customers with active subscriptions to any of the following service platforms can create developer databases:

- Autonomous AI Database on Dedicated Exadata Infrastructure
- Oracle Exadata Database Service on Dedicated Infrastructure
- Autonomous AI Database on Exadata Cloud@Customer
- Oracle Exadata Database Service on Cloud@Customer

There is no limit on the number of free developer databases; it's limited by the capacity of your Exadata infrastructure.

Supported Oracle Database Versions

Autonomous AI Database for Developers can be created with *Oracle Database 19c* or *Oracle Database 26ai*, depending on the database software version of its parent Autonomous Container Database (ACD). For example, to create a developer database with Oracle Database 26ai, you must choose an ACD whose Oracle database software version is 26ai.

To know more about each database version, see Oracle Database 26ai or [Oracle Database 19c](#) on *Oracle Help Center*.

Provisioning Workflow

You can provision an Autonomous AI Database for Developers from the Oracle Cloud Infrastructure (OCI) console or using API.

To create a developer database, you need an Autonomous Container Database (ACD) without an Autonomous Data Guard in an ECPU-based Autonomous Exadata VM Cluster (AVMC).

If you do not have these resources provisioned already, create the ECPU-based AVMC first and then create an ACD without disaster recovery (Autonomous Data Guard) using that AVMC.

After creating or identifying (if they already exist) the AVMC and ACD, you can create an Autonomous AI Database for Developers using them. Provisioning a developer database using the OCI console follows the same workflow as creating a regular Autonomous AI Database, as described in [Create an Autonomous AI Database](#). Once created, the Autonomous AI Database for Developers instances appear with a **Developer label** in the list of Autonomous AI Databases on the OCI console.

Specifications

Each developer database comes with the following specifications:

- **Compute:** Fixed 4 ECPUs, with no CPU scaling
- **Storage:** Fixed 20 GB
- **Session limits:** 30 concurrent database sessions
- **Workload type:** Autonomous AI Lakehouse, Autonomous AI Transaction Processing

Excluded Features

Autonomous AI Database for Developers supports all the features offered by a regular Autonomous AI Database except those listed below. These limitations are in place to ensure that the developer databases are optimally used as a development sandbox.

Developer database instances:

- Do not support Autonomous Data Guard. Hence, they can only be provisioned in an ACD without Autonomous Data Guard.
- Support ECPU only. Therefore, you can provision them only on an ECPU based ACD.
- Come with fixed compute and storage sizing, do not support manual or auto-scaling and storage scaling.
- Can not have long-term backups.
- Do not provide Database In-memory.

Supported Features

- **Cloning:** The Autonomous AI Database for Developers offers fewer resources and features than the regular Autonomous AI Database. For non-development uses, such as load/stress testing and production, or to access all features, users can clone from a developer database to a regular Autonomous AI Database.
You can also clone a regular database into a developer database. However, to successfully clone a regular database into a developer database, the actual used space of the source database, rounded up to the next GB, *must be 32GB or less*.

See [Clone an Autonomous AI Database](#) for instructions.

- **Backup and Recovery:** You can enable automatic backups or trigger manual backups of your developer database as needed. If the backup destination is an Object Storage and Recovery Service, the backups will be billed.

- **Service Maintenance:** Developer databases follow the same patching schedule as regular Autonomous AI Database; however, they will not support critical one-off patches.
- **Database Application Development and Developer Tools:** With Autonomous AI Database for Developers, you can use all the developer-related features and built-in tools that an Autonomous AI Database offers. See *Develop Applications with Autonomous AI Database* in *Tasks* and *Service Monitoring* in *Features* for complete details.

Autonomous AI Database for Developers comes with a service level objective (SLO) of 99.5%, and you can log service requests (SRs) to Oracle Support for assistance. However, there is no Severity 1 SR support for developer databases. See *Create a Service Request in My Oracle Support* to learn how to contact Oracle Support for assistance.

Billing Autonomous AI Database on Dedicated Exadata Infrastructure

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure uses specific algorithms to allocate and bill for usage of the compute used by Autonomous AI Databases. Understanding these algorithms can help you determine how best to create and configure your Autonomous AI Databases to meet performance goals in the most cost-effective fashion.

CPU Billing Details

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure computes CPU billing as follows:

1. CPU usage for each Autonomous AI Database is measured each second in units of whole ECPU or OCPU.
 - a. A stopped Autonomous AI Database uses zero ECPU or OCPU. When an Autonomous AI Database is stopped, you are not billed.
 - b. A running Autonomous AI Database uses its allocated number of ECPUs or OCPUs plus any additional ECPUs or OCPUs due to auto-scaling. When an Autonomous AI Database is running, you are billed for the number of CPUs currently allocated to the database, whether specified at initial creation or later by a manual scaling operation. Additionally, if auto-scaling is enabled for the database, you are billed for each second for any additional CPUs the database is using as the result of being automatically scaled up.

Note

Creating AVMC and ACD resources does not initiate billing. So, even though you assign a total CPU count to an AVMC and each ACD consumes 8 ECPUs or 2 OCPUs per node when created, these CPUs are not billed. Only once you provision Autonomous AI Databases in an AVMC and an underlying ACD, and that database is actively running, will the CPUs used be billed. As a result, you can create ACDs within AVMCs to organize and group your databases according to your lines of business, functional areas, or some other technique without worrying about incurring costs.

- c. When you create an Autonomous AI Database, by default Oracle reserves additional CPUs to ensure that the database can run with at least 50% capacity even in case of any node failures. You can change the percentage of CPUs reserved across nodes to 0% or 25% while provisioning an ACD. See **Node failover reservation** in *Create an*

Autonomous Container Database for instructions. These additional CPUs are not included in the billing.

Note

Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@AWS supports ECPU compute model only.

2. The per-second measurements are averaged across each hour interval for each Autonomous AI Database.
3. The per-hour averages for the Autonomous AI Databases are added together to determine the CPU usage per hour across the entire Autonomous VM Cluster resource.

Autonomous AI Database on Dedicated Exadata Infrastructure database compute costs are aggregated and reported at the AVMC level, covering all active Autonomous AI Databases across all ACDs in the AVMC. OCI Cost Analysis can provide the AVMC's usage and cost.

To estimate the cost per Autonomous AI Database, sum the total ECPUs across the Autonomous AI Databases and allocate cost based on each Autonomous AI Database's share of total CPU consumption. For example:

If the AVMC reports **1500 ECPUs billed** for a billing period and three Autonomous AI Databases are active; Database A with **10 ECPUs**, Database B with **20 ECPUs**, and Database C with **30 ECPUs**. The cost split is:

- **Database A:** $10/(10+20+30) = 16.67\%$ of 1500
- **Database B:** $20/(10+20+30) = 33.33\%$ of 1500
- **Database C:** $30/(10+20+30) = 50\%$ of 1500

This assumes fixed CPU sizes with no auto-scaling, and all 3 Autonomous AI Databases were running during the billing period. For greater accuracy, use the `ECPUsAllocated` metric to capture actual ECPU usage per Autonomous AI Database.

It is recommended that you see:

- [Compute Management in Autonomous AI Database](#) to learn how CPUs move between total, available, and reclaimable CPU categories with usage and how they are billed.
- [CPU Allocation When Auto-Scaling](#) to understand how the CPUs allocated with auto-scaling impact billing, with specific examples.
- [Cost Analysis](#) to visualize, track, and optimize your Oracle Cloud Infrastructure spending.

Elastic Pool Billing

An elastic pool allows you to consolidate your Autonomous AI Database instances in terms of their compute resource billing.

You can think of an elastic pool like a mobile phone service "family plan," except this applies to your Autonomous AI Database instances. Instead of paying individually for each database, the databases are grouped into a pool in which one instance, the leader, is charged for the compute usage associated with the entire pool. See [Consolidate Autonomous AI Database Instances Using Elastic Pools](#) for complete details about elastic resource pools.

Elastic resource pool usage:

- Is billed to the pool leader, and billing is based on the elastic resource pool size and the actual hourly ECPU usage of the pool leader and the members.

- Can exceed the pool size (pool capacity can be up to four times greater than the pool size).
- Billing consists of only compute resources, that is, ECPU usage, and all compute usage is charged to the Autonomous AI Database instance that is the pool leader.

Using an elastic pool, you can provision up to four times the number of ECPUs over your selected pool size, and you can provision database instances that are in the elastic pool with as little as 1 ECPU per database instance. Outside of an elastic pool the minimum number of ECPUs per database instance is 2 ECPUs. For example, with a pool size of 128, you can provision 512 Autonomous AI Database instances (when each instance has 1 ECPU). In this example, you are billed for the pool size compute resources, based on the pool size of 128 ECPUs, while you have access to 512 Autonomous AI Database instances. In contrast, when you individually provision 512 Autonomous AI Database instances without using an elastic pool, you must allocate a minimum of 2 ECPUs for each Autonomous AI Database instance. In this example, you would pay for 1024 ECPUs. Using an elastic pool provides **up to 87% compute cost savings**.

After creating an elastic pool, the total ECPU usage for a given hour is charged to the Autonomous AI Database instance, that is the pool leader. Except for the pool leader, individual Autonomous AI Database instances that are pool members are not charged for ECPU usage while they are members of an elastic pool.

Elastic pool billing is as follows:

- If the total aggregated peak ECPU utilization is equal to or below the pool size for a given hour, you are charged for the pool size number of ECPUs (one times the pool size).
- After an elastic pool is created, ECPU billing continues at a minimum of one-time the pool size rate, even when pool member databases and the pool leader are stopped.
- In other words, if the aggregated peak ECPU utilization of the pool is less than or equal to the pool size for a given hour, you are charged for the pool size number of ECPUs (one times the pool size). This represents **up to 87% compute cost savings** over the case in which these databases are billed separately without using elastic pools.
- If the aggregated peak ECPU utilization of the pool leader and the members exceeds the pool size at any point in time in a given billing hour:
 - **Aggregated peak ECPU utilization of the pool is equal to or less than two times the pool size number of ECPUs:** For usage that is greater than one times the pool size number of ECPUs and up to and including two times the number of ECPUs in a given billing hour: Hourly billing is two times the pool size number of ECPUs. In other words, if the aggregated peak ECPU utilization of the pool exceeds the pool size but is less than or equal to two times the pool size for a given hour, you are charged twice the pool size number of ECPUs (two times the pool size). This represents up to 75% compute cost savings over the case in which these databases are billed separately without using elastic pools.
 - **Aggregated peak ECPU utilization of the pool is equal to or less than four times the pool size number of ECPUs:** For usage that is greater than two times the pool size number of ECPUs and up to and including four times the pool size number of ECPUs in a given billing hour, hourly billing is four times the pool size number of ECPUs. In other words, if the aggregated peak ECPU utilization of the pool exceeds twice the pool size for a given hour, you are charged for four times the pool size number of ECPUs (four times the pool size). This represents **up to 50% compute cost savings** over the case in which these databases are billed separately without using elastic pools.
- For example, consider an elastic pool with a pool size of 128 ECPUs and a pool capacity of 512 ECPUs:

- **Case-1:** The aggregated peak ECPU utilization of the pool leader and the members is 40 ECPUs between 2:00 pm and 2:30 pm and 128 ECPUs between 2:30 pm and 3:00 pm.
 - * The elastic pool is billed 128 ECPUs, one-time the pool size, for this billing hour (2–3 pm). This case applies when the peak aggregated ECPU usage of the elastic pool for the billing hour is less than or equal to 128 ECPUs.
- **Case 2:** The aggregated peak ECPU utilization of the pool leader and the members is 40 ECPUs between 2:00 pm and 2:30 pm and 250 ECPUs between 2:30 pm and 3:00 pm.
 - * The elastic pool is billed 256 ECPUs twice the pool size for this billing hour (2–3 pm). This case applies when the peak aggregated ECPU usage of the elastic pool for the billing hour is less than or equal to 256 ECPUs and greater than 128 ECPUs.
- **Case-3:** The aggregated peak ECPU utilization of the pool leader and the members is 80 ECPUs between 2:00 pm and 2:30 pm, and 509 ECPUs between 2:30 pm and 3:00 pm.
 - * The elastic pool is billed 512 ECPUs, four times the pool size, for this billing hour (2–3 pm). This case applies when the peak aggregated ECPU usage of the elastic pool for the billing hour is less than or equal to 512 ECPUs and greater than 256 ECPUs.

For more details, see [How to Achieve up to 87% Compute Cost Savings with Elastic Resource Pools on an Autonomous AI Database](#).

Elastic Pool Billing when a Pool is Created or Terminated

When an elastic pool is created or terminated, the leader is billed for the entire hour for the elastic pool. In addition, individual instances that are either added or removed from the pool are billed for any compute usage that occurs while the instance is not in the elastic pool (in this case, the billing applies to the individual Autonomous AI Database instance).

- **Pool Creation Example:** Assume an Autonomous AI Database instance with 4 ECPUs is not part of any elastic pool. At 2:15 pm, if you create an elastic pool with this instance with a pool size of 128 ECPUs, the instance becomes a pool leader. Assuming the Autonomous AI Database idles between 2–3 pm, and there are no other Autonomous AI Database instances in the pool, billing for the hour between 2–3 pm is as follows:
 - The bill for the period 2–3 pm is: $(4 * 0.25) + 128 = 129$ ECPUs.
 - Where $(4 * 0.25)$ is the billing for compute for the fifteen minutes before the Autonomous AI Database instance created the elastic pool (during which period from 2 to 2:15 pm, the instance will be billed against the VM cluster), and 128 ECPUs is the billing for the elastic pool for the hour when it is created.
- **Pool Termination Example:** Assume an Autonomous AI Database instance with 4 ECPUs is the leader of an elastic pool, and the pool size is 128 ECPUs. At 4:30 pm, if you terminate the elastic pool, the database becomes a standalone Autonomous AI Database instance, not part of any elastic pool. Assuming the Autonomous AI Database idles between 4–5 pm, and there are no other Autonomous AI Database instances in the pool, billing for the hour between 4–5 pm is as follows:
 - The bill for 4–5 pm is: $(4 * 0.5) + 128 = 130$ ECPUs.
 - Where the $(4 * 0.5)$ is the billing for compute for the thirty minutes after the Autonomous AI Database instance terminates the elastic pool, and 128 ECPUs is the billing for the elastic pool for the hour when the elastic pool was terminated.

- Once the Autonomous AI Database instance leaves the pool, it becomes part of the VM cluster again, and gets billed against the VM cluster.

Elastic Pool Billing when a Pool Member or Leader Leaves the Pool

Billing for an Autonomous AI Database instance that leaves an elastic pool returns to individual instance billing based on the compute resources that the individual instance uses:

- If a pool member with 2 ECPUs or more leaves the pool, the individual instance's ECPU allocation remains, and the instance is billed for that number of ECPUs.
- If a pool member with 1 ECPU leaves the pool, the ECPU allocation is automatically set to 2 ECPUs, and the instance is billed for 2 ECPUs going forward unless it's scaled up.

2

Quick Start

Before You Begin with Autonomous AI Database

Before you begin using Autonomous AI Database, you should be familiar with Oracle Cloud. See *Getting Started with Oracle Cloud*.

Before you create Autonomous AI Database on Dedicated Exadata Infrastructure deployments:

- On Oracle Cloud, purchase a subscription. You cannot create an Autonomous AI Database on Dedicated Exadata Infrastructure deployment until you do so.
- Ensure your service limits show at least one supported Exadata system shape available. Before attempting to create the infrastructure resources, request a service limit increase if necessary. See [Request Service Limit Increase](#) for guidance.
- Institute access controls to ensure that only authorized personnel and applications have network access to the Autonomous AI Databases created on Dedicated Exadata Infrastructure. See [Add Users, Assign Policies and Roles](#) for more information.
- In addition, if you are a customer with a Multicloud subscription, see [Prerequisites for Oracle Database@AWS](#) or [Prerequisites for Oracle Database@Azure](#) for more information, depending on your Cloud Service Provider.
- To get started with Autonomous AI Database on Dedicated Exadata Infrastructure, the fleet administrator must create the core infrastructure resources listed below:
 - First create an **Exadata Infrastructure (EI)** resource to host the Autonomous AI Database resources. Depending on the platform that you are deploying on, choose one of the following options:
 - * Deploying on Oracle Public Cloud:
 - * [Characteristics of Infrastructure Shapes](#)
 - * [Resource Limits](#)
 - * [Create an Exadata Infrastructure Resource](#)
 - * Deploying on Exadata Cloud@Customer:
 - * [Preparing for Exadata Cloud@Customer](#)
 - * [Provisioning Exadata Cloud@Customer Systems](#)
 - * [Create an Exadata Infrastructure Resource](#)
 - * Deploying on Multicloud:\
 - * Oracle Database @AWS:
 - * [Service Limits for Oracle Database@AWS](#)
 - * [Create an Exadata Infrastructure in AWS region](#)
 - * Oracle Database@Azure:
 - * [Service Limits for Oracle Database@Azure](#)

- * [Create an Exadata Infrastructure in Azure region](#)
- Create an **Autonomous Exadata VM Cluster (AVMC)** resource on the Exadata Infrastructure. Depending on the platform that you are deploying on, choose one of the following options:
 - * Oracle Public Cloud or Exadata Cloud@Customer:
 - * Create an Autonomous Exadata VM Cluster
 - * Multicloud:
 - * Oracle Database@AWS:
 - * [Create an Autonomous Exadata VM Cluster in AWS region](#)
 - * Oracle Database@Azure
 - * [Create an Autonomous Exadata VM Cluster in Azure Region](#)
- Create at least one **Autonomous Container Database (ACD)** in the AVMC resource following the instructions in Create an Autonomous Container Database.
- You can now create an **Autonomous AI Database** using the infrastructure resources created above. See Create an Autonomous AI Database for details.
- Optionally, if you want to leverage Oracle Cloud Infrastructure Object Storage for data loading, you need your object store credentials to use with Autonomous AI Database on Dedicated Exadata Infrastructure, including a username and a password. Your username is your Oracle Cloud Infrastructure user name, and your password is your auth token. See [Working with Auth Tokens](#) for more details.

After you create Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can connect to your Autonomous AI Database and develop applications. Refer to Develop Applications with Autonomous AI Database on Dedicated Exadata Infrastructure for guidance.

Note

For Oracle Public Cloud and Exadata Cloud@Customer deployments, you can refer to the following use cases for a demonstration of the above discussed steps with an example:

- Configure Autonomous AI Database for Proof of Concept (POC)
- Configure Autonomous AI Database with Reference Architecture

Request Service Limit Increase

Before creating an Exadata Infrastructure resource, you are recommended to check the available service limits and request a service limit increase from the Oracle Cloud Infrastructure console if required.

You need to have a minimum number of Exadata Database Servers and Exadata Storage Servers available in your tenancy to provision an Exadata Infrastructure. These service limits are determined by the Exadata System Shape that you choose to provision your Exadata Infrastructure.

The Exadata System Database Server Count and Storage Server Count limits are defined as counts per availability domain.

If you try to create a resource for which a limit has been met, you're prompted to submit a limit increase request.

To Request a Service Limit Increase

If you do not have the required service limits to provision an Exadata Infrastructure using the Exadata System Shape of your choice, you can submit a request to increase your service limits using the Oracle Cloud Infrastructure console to add Exadata Infrastructure resources - Database Servers and Storage Servers to the tenancy.

Note

This procedure applies to requests for service limit increases. For details about the subscribed region limit and how to request an increase to that limit, see [Subscribed Region Limits](#).

To view the current service limits and request a service limit increase:

1. From the OCI Console, open the navigation menu and click **Governance & Administration**.
2. Under **Support**, click **Request Service Limit Updates**.
3. On the **Limit Increase** tab, select *Database* for **Service category**.
4. Select a service limit specific to the Exadata System Shape of your choice from the **Resource** select list.

For example, the service limits applicable to X9M-2 Exadata System on Oracle Public Cloud are **Exadata X9M Database Server Count - 126 Cores** and **Exadata X9M Storage Server Count - 64 TB** which must be set to 2 and 3 respectively, at the least.

5. The service limits and current usage is displayed *per Availability Domain (AD)*.

Note

Observe that the Exadata System Database Server Count and Storage Server Count limits are defined as counts *per availability domain*.

6. If you do not have enough service limits, increase the service limit value for one or more ADs listed.

The minimum required service limits are determined by the Exadata System Shape that you choose use. See Characteristics of Infrastructure Shapes for a complete list of Exadata Systems supported by Autonomous AI Database on Dedicated Exadata Infrastructure and their specifications.

7. The **Limit Increase** tab shows one service limit entry per resource. So, to include another resource in this request, you must click **+ Additional Request** and follow the same procedure to increase its current limits.

Extending the previous example, if you selected **Exadata X9M Database Server Count - 126 Cores** earlier and want to increase the limits for Storage Server Count also, then you must click **+ Additional Request** and select **Exadata X9M Storage Server Count - 64 TB** for resource. This displays the service limit and current usage for the Storage Server Count as a second row.

Request Service Limit Updates

Submit a request to update service limits for resources in your tenancy. If you are trying to change the **compartment quota** for a resource, your **administrator** can increase the quota.

Resource Limit Update (this will be your new limit)

Service Category: **Database** Resource: **Exadata XDM Database Server Count - 126 Cores**

Limits for this resource type are defined as a count per availability domain.

ztr US-ASHBURN-AD-1 Limit	ztr US-ASHBURN-AD-2 Limit	ztr US-ASHBURN-AD-3 Limit
2	2	0
Service limit 2. Current usage 2.	Service limit 0. Current usage 0.	Service limit 0. Current usage 0.

Resource Limit Update (this will be your new limit)

Service Category: **Database** Resource: **Exadata XDM Storage Server Count - 64 TB**

Limits for this resource type are defined as a count per availability domain.

ztr US-ASHBURN-AD-1 Limit	ztr US-ASHBURN-AD-2 Limit	ztr US-ASHBURN-AD-3 Limit
3	3	0
Service limit 3. Current usage 3.	Service limit 0. Current usage 0.	Service limit 0. Current usage 0.

+ Additional Request

Create Support Request **Cancel**

8. Optionally, enter a **Reason for request**, and click **Create support request**

The requested service limits are usually allocated within 1 business day, but may take longer depending on the availability and the quantity requested.

If you need your request to be handled urgently, you can explain the severity of your situation in the **Reason for Request** text box.

Access Autonomous AI Database in the Oracle Cloud Infrastructure Console

You can access a Autonomous AI Database from the Oracle Cloud Infrastructure console.

1. Open your web browser and go to <https://cloud.oracle.com>.
2. Enter the name of your cloud account in the **Account** field and then click **Next**.
3. On the **Cloud Infrastructure** sign-in page, enter your sign-in credentials under **Oracle Cloud Infrastructure Direct Sign-In**. Then click **Sign In**.
4. Click the  menu icon in the top corner to display the navigation menu.
5. Click **Oracle AI Database** in the navigation menu.
6. Choose an item under Oracle AI Database based on whether your dedicated infrastructure resources are on Oracle Cloud or Oracle Exadata Cloud@Customer:
 - On **Oracle Public Cloud**, Click **Autonomous AI Database on Dedicated Infrastructure**.
The **Autonomous AI Databases** page opens, showing the list of databases in your current Compartment.
 - On **Oracle Exadata Cloud@Customer**, Click **Oracle Exadata Database Service on Cloud@Customer**.
The **VM Clusters** page opens. On this page, click the type of Autonomous AI Database resource you want to manage.

Obtain Tenancy Details

When you file a service request or when you need Autonomous AI Database tenancy details for other reasons you can obtain these details for your instance. Tenancy details for an instance are available on the Oracle Cloud Infrastructure Console or you can obtain these details by querying the database.

If you need to file a service request use [Oracle Cloud Support](#) or contact your support representative and provide the first five of the following items (service requests do not require that you supply the outbound IP address or the availability domain):

- [Database Name](#)
- [Region](#)
- [Tenancy OCID](#)
- [Database OCID](#)
- [Compartment OCID](#)
- [Outbound IP Address](#)
- [Availability Domain](#)

If you are connected to the database you can obtain tenancy details by querying the CLOUD_IDENTITY column of the V\$PDBS view.

For example, running the following:

```
SELECT cloud_identity FROM v$pdb;
```

```
CLOUD_IDENTITY
```

```
-----
{
  "DATABASE_NAME" : "DBxxxxxxxxxxxx",
  "REGION" : "us-phoenix-1",
  "TENANT_OCID" : "OCID1.TENANCY.REGION1..ID1",
  "DATABASE_OCID" : "OCID1.AUTONOMOUSDATABASE.OC1.IAD.ID2",
  "COMPARTMENT_OCID" : "ocid1.tenancy.region1..ID3"
  "OUTBOUND_IP_ADDRESS" :
  [
    "192.0.2.254"
  ]
  "AVAILABILITY_DOMAIN" : "SoSC:region-1-AD-1"
}
```

If your Autonomous AI Database instance was created before the tenancy details feature was added and has not been restarted, then this query does not return tenancy details. In this case, as a one-time operation, restart your instance and run the query again. You can restart your instance from the Oracle Cloud Infrastructure console or using the restart API.

Public IP Address Ranges for Autonomous AI Database

Oracle Cloud Infrastructure provides information about public IP address ranges for Autonomous AI Databases that are deployed in Oracle Cloud Infrastructure.

See [IP Address Ranges](#) for more information.

Database Name

You set the database name when you provision a database or when you rename a database.

The Oracle Cloud Infrastructure Console shows the database name on the Autonomous AI Database Details page under General Information in the **Database Name** field.

Region

The Oracle Cloud Infrastructure Console shows the region on the console, in the Regions area.

Tenancy OCID

The tenancy details page shows the tenancy OCID.

See [Managing the Tenancy](#) for information on accessing the Tenancy Details page.

See [Resource Identifiers](#) for information on Oracle Cloud Identifiers.

Database OCID

The Oracle Cloud Infrastructure Console shows the database OCID on the Autonomous AI Database Details page under General Information in the **OCID** field.

See [Resource Identifiers](#) for information on Oracle Cloud Identifiers.

Compartment OCID

See [Managing Compartments](#) for more information on compartments.

See [Resource Identifiers](#) for information on Oracle Cloud Identifiers.

Outbound IP Address

The `OUTBOUND_IP_ADDRESS` is the outbound IP address of your Autonomous AI Database instance. You can use the `OUTBOUND_IP_ADDRESS` when you create a database link to another Autonomous AI Database instance that uses ACLs to restrict access. In this case, you need to allow the specified outbound IP address to connect to the target Autonomous AI Database and then create the database link.

The `OUTBOUND_IP_ADDRESS` shows the outbound IP address in the following cases:

- When your Autonomous AI Database instance uses a Public Endpoint.
- When your Autonomous AI Database instance uses a Private Endpoint and the database property `ROUTE_OUTBOUND_CONNECTIONS` is set to `' '` (the default value).

Note

When your Autonomous AI Database instance uses a Private Endpoint and you set the database property `ROUTE_OUTBOUND_CONNECTIONS` to `'PRIVATE_ENDPOINT'`, outbound connections go through the private endpoint.

See [Create One-way TLS Database Links from an Autonomous AI Database to another Autonomous AI Database](#) for information on creating database links.

See [Client Access Control](#) for information on configuring access control rules.

Availability Domain

You can view the Availability Domain of your Autonomous AI Database instance.

This is useful when you need to deploy your Application Servers when your Autonomous AI Database instance resides in a multi-Availability Domain region. In order to reduce latency as much as possible you would want to deploy your Application Server and database in the same Availability Domain.

See [Regions and Availability Domains](#) for more information on Availability Domains.

Get Help and Contact Oracle Support

When using Autonomous AI Database on Dedicated Exadata Infrastructure, there will be times when you need to get help from the community or to talk to someone in Oracle support.

In such a situation, you can submit a question to Cloud Customer Connect at cloudcustomerconnect.oracle.com, or you can create a service request to engage the Oracle Support team.

Create a Service Request in My Oracle Cloud Support

If you are having a technical issue using Autonomous AI Database on Dedicated Exadata Infrastructure, you can contact Oracle Support by creating a service request in My Oracle Cloud Support.

If you are an Oracle Commercial Cloud customer, and you need the ability to request support for a tenancy that your cloud administrator created *after November 8th, 2024*, then you need to sign into My Oracle Cloud Support using **Sign in with your commercial cloud account**, and enter the new tenancy credentials. If you are unsure about your tenancy credentials, please contact your cloud administrator.

Tip

Before following these steps, sign in to your Oracle Cloud account at cloud.oracle.com in a separate browser tab or window so that you can quickly and easily get any information you need to complete your service request.

- [Oracle Commercial Cloud customer](#)

- [Oracle Non-Commercial Cloud Customer](#)

Oracle Commercial Cloud customer

If you are an Oracle Commercial Cloud customer and you know your cloud account login information:

1. Go to support.oracle.com.
2. Enter your *email address* and click **Next**.
 - If you see a message that your email address is not recognized, follow the instructions outlined in [Register to My Oracle Cloud Support as a New User](#).
 - If your email is registered with Oracle Support, you'll be directed to sign in to the applicable support portal for your products and services. Continue with step 3.
3. Click **Sign in with your commercial cloud account**.
4. Enter your *OCI credentials* (Tenancy, domain if applicable, username, and password).

If this is your first time signing in to My Oracle Cloud Support, you will be prompted to associate your cloud account to My Oracle Cloud Support by verifying your email.
5. Accept the **Terms of Use**.
6. Complete your **personal Info**.
 - For **Security Pin**, enter a 6 digit number not starting with a 0. When you call Oracle support line, you will be asked to provide this pin.
 - If this is the first time you are filling the **personal Info**, you may have to verify your **Mobile Phone** using a one-time password sent to your mobile number.
7. Click **Save** and you will be routed to the **My Oracle Cloud Support** portal.
8. To create a new service request, click **Create Service Request**.
9. Provide the details prompted on the **Create a new service request** page, clicking **Next**, as needed.
10. When you see **Proceed with the next step to submit the SR.**, click **Next**.

Oracle Non-Commercial Cloud Customer

If you are an Oracle Cloud customer and you do not have cloud account login information OR you have a non-commercial cloud account (government realm or dedicated realm):

1. Go to support.oracle.com.
2. Enter your *Oracle account username and password*.
3. If this your first time signing in to My Oracle Cloud Support, you will be prompted to accept the **Terms of use**, and complete your *personal profile*. Refer to [Register to My Oracle Cloud Support as a New User](#) for more details.
4. Once you get access to My Oracle Cloud Support, sign in to support.oracle.com.
5. You are routed to the **My Oracle Cloud Support** portal.
6. To create a new service request, click **Create Service Request**.
7. Provide the details prompted on the **Create a new service request** page, clicking **Next**, as needed.
8. When you see **Proceed with the next step to submit the SR.**, click **Next**.

Note

If you are the first user logging in using a specific Subscription ID and Account name, you will be prompted to accept the Administrator role.

Register to My Oracle Cloud Support as a New User

- [Oracle Commercial Cloud customer](#)
- [Oracle Non-Commercial Cloud Customer](#)

Oracle Commercial Cloud customer

If you are an Oracle Commercial Cloud customer:

1. Go to support.oracle.com.
2. Enter your *email address* and click **Next**.
3. Click **Sign in** with your commercial cloud account.
4. Enter your *Tenancy*, then click **Continue**.
If you do not know your tenancy information, please contact your cloud administrator.
5. Choose an **Identity domain** if applicable, then click **Next**.
6. Enter your *cloud account user name* and *password*.
7. Associate your cloud account to My Oracle Cloud Support by verifying your email.
8. Accept **Terms of Use**.
9. Complete your **personal Info**.
 - For **Security Pin**, enter a 6 digit number not starting with a 0. When you call Oracle support line, you will be asked to provide this pin.
 - If this is the first time you are filling the **personal Info**, you may have to verify your **Mobile Phone** using a one-time password sent to your mobile number.
10. You will be directed to the My Oracle Cloud Support profile page as an **approved user**.

Oracle Non-Commercial Cloud Customer

If you are an Oracle Cloud customer and you do not have cloud account login information OR you have a non-commercial cloud account (government realm or dedicated realm):

1. Click **Create an Oracle Account**.
2. In **Create Your Oracle Account**, provide the required information and click **Create Account**.
3. You will receive a verification email to enable your new account. Once your email has been verified, you will be prompted to log in with your newly created Oracle account username and password.
4. Enter *subscription ID* and *Account or Company name* to complete identity verification.

Contact your Cloud administrator for the subscription ID and Account name.

5. Complete your **personal Info**.
 - For **Security Pin**, enter a 6 digit number not starting with a 0. When you call Oracle support line, you will be asked to provide this pin.
 - If this is the first time you are filling the **personal Info**, you may have to verify your **Mobile Phone** using a one-time password sent to your mobile number.
6. You will be directed to the My Oracle Cloud Support profile page as a **pending user**. If approval is not granted within a reasonable amount of time, contact your administrator. Once you are approved, you will have access to My Oracle Cloud Support.

Note

The first user logging in using a Subscription ID and Account name will be prompted to accept the Admin or CUA role.

Create a Support Ticket from Oracle Cloud Infrastructure console

You can also create a support ticket from the Oracle Cloud Infrastructure console.

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

To create a support request from the Oracle Cloud Infrastructure Console, follow the instructions in [Creating a Support Request](#).

LiveLabs for Autonomous AI Database on Dedicated Exadata Infrastructure

To try out Autonomous AI Database on Dedicated Exadata Infrastructure features, you can view the workshops offered on Oracle LiveLabs. These workshops provide step-by-step instructions to set up and use your Autonomous AI Database on Oracle Public Cloud and Exadata Cloud@Customer.

Note

You need Oracle Cloud Infrastructure credentials to try out the labs in these workshops.

To access the Autonomous AI Database workshops:

- Click Oracle LiveLabs, and search for *Autonomous AI Database* in the **Search Bar**.
Oracle LiveLabs offers the following workshops for Autonomous AI Database, catering to different user roles.
 1. Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop

2. Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop
 3. Oracle Autonomous AI Database Dedicated for Security Administrators Workshop
 4. Oracle Database@AWS- Autonomous AI Database on Dedicated Infrastructure
- Click **Launch** on the tile you are interested in, and click **Run On Your Tenancy** to launch the workshop.

Workflows

Set up Infrastructure Resources for Autonomous AI Database on Dedicated Exadata Infrastructure

To get started with Autonomous AI Database on Dedicated Exadata Infrastructure, the fleet administrator must create the core infrastructure resources and institute access controls to ensure that only authorized personnel and applications have network access to the Autonomous AI Databases created on dedicated Exadata infrastructure.

Task	Description	More Information
Determine and institute user access constraints	Determine the controls you deem necessary to ensure that your users have the appropriate access to the appropriate cloud resources to perform their job duties, and then create infrastructure resources to institute these controls.	Plan Access Controls and Create Supporting Resources
Create core infrastructure resources	Create an Exadata Infrastructure resource and then create an Autonomous Exadata VM Cluster resource on it. If you are deploying on Oracle Cloud, make sure your service limits show at least one supported Exadata system shape available. Before attempting to create the infrastructure resources, request a service limit increase if necessary.	Deploying on Oracle Public Cloud: • Service Limits in <i>Oracle Cloud Infrastructure Documentation</i> • Characteristics of Infrastructure Shapes • Resource Limits • Create an Exadata Infrastructure Resource • Create an Autonomous Exadata VM Cluster Deploying on Exadata Cloud@Customer: • Preparing for Exadata Cloud@Customer • Provisioning Exadata Cloud@Customer Systems • Create an Autonomous Exadata VM Cluster Deploying on Multicloud: • Oracle Database@AWS: – Service Limits for Oracle Database@AWS – Create an Exadata Infrastructure in AWS region – Create an Autonomous VM Cluster (AVMC) • Oracle Database@Azure: – Service Limits for Oracle Database@Azure – Create an Exadata Infrastructure in Azure Region

Task	Description	More Information
Create Autonomous Container Database resources	Create at least one Autonomous Container Database in an Autonomous Exadata VM Cluster resource.	Create an Autonomous Container Database
Create Autonomous AI Database	Create at least one Autonomous AI Database in an Autonomous Container Database resource.	Create a Dedicated Autonomous AI Database

Note

For Oracle Public Cloud and Exadata Cloud@Customer deployments, you can refer to the following use cases for a demonstration of the above discussed steps with an example:

- Configure Autonomous AI Database for Proof of Concept (POC)
- Configure Autonomous AI Database with Reference Architecture

Create and Use Autonomous AI Database

You can access your Autonomous AI Database, connect it to an application, load data, and start using it after your fleet administrator sets up the infrastructure resources and defines access controls and an application database administrator (Application DBA) creates an Autonomous AI Database with database users.

Task	Description	More Information
Create and log in to your cloud account	Provide your information and sign up for an Oracle Cloud Service.	Getting Started with Oracle Cloud
Provision Autonomous AI Database	Create an Autonomous AI Database instance.	Create a Dedicated Autonomous AI Database
Start the instance	Start and verify that an Autonomous AI Database is up and running.	Start a Dedicated Autonomous AI Database
Create database users and obtain security credentials	Create accounts for your users and assign them appropriate privileges. Assign the necessary Autonomous AI Database roles.	Create Database Users
Connect to your database	Obtain security credentials and connect to your Autonomous AI Database using a database client or tool.	Connect to a Dedicated Autonomous AI Database
Scale the database	Use the Oracle Cloud Infrastructure console to scale an instance to add or remove CPU or storage resources.	Add CPU or Storage Resources to a Dedicated Autonomous AI Database
Load data into the database	Use the available tools to load data into the database.	Loading Data into Autonomous AI Database
Monitor the database	Check on the health and performance of the database.	Manage and Monitor Performance

Task	Description	More Information
Manage the database	Check on the day-to-day operation of your service, monitor performance, and review important notifications.	All aspects of the database lifecycle are simplified by the service, including patching and upgrading the database. You can obtain service information including the service and database version from the service console. Back Up and Restore Autonomous AI Database

Develop Applications with Autonomous AI Database

You can connect to your Autonomous AI Database and start developing database applications after your fleet administrator sets up infrastructure resources and an application database administrator (Application DBA) provisions an Autonomous AI Database with database users.

Task	Description	More Information
Create an Autonomous AI Database	Create the database that your application will use.	Create the Database
Configure a development system	Ensure that the development system you are using has network access to the database you want to connect to.	Configure a Development System to Access the Database
Connect to your database	Obtain security credentials and connect to your Autonomous AI Database using a database client or tool.	Connect to a Dedicated Autonomous AI Database
Define or load the database structure	Define the database's tables, views, triggers, types, sequences and so on.	Define or Load the Database Structure
Begin building an application	Start building an application that takes advantage of the high-performance and continuous-availability features of an Autonomous AI Database.	The Basics of Building an Application

3

Features

High Availability

Protect Critical Databases from Failures and Disasters Using Autonomous Data Guard

The Autonomous Data Guard feature enables you to keep your critical production databases available to mission critical applications despite failures, disasters, human error, or data corruption. This kind of capability is often called *disaster recovery*.

In Autonomous AI Database on Dedicated Exadata Infrastructure, you configure and manage Autonomous Data Guard at the Autonomous Container Database level.

About Autonomous Data Guard

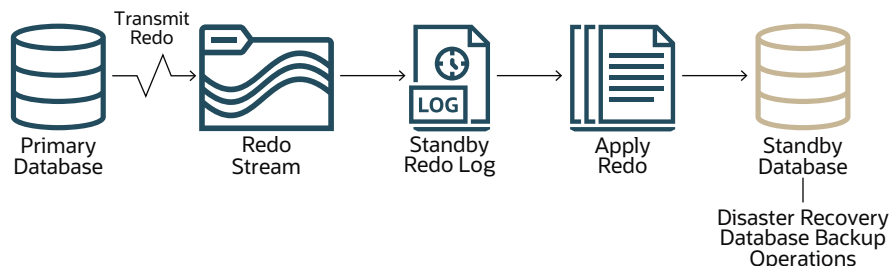
Autonomous Data Guard creates and maintains two completely separate copies of your database: a *primary* database that your applications connect to and use, and a *standby* database that is a synchronized copy of the primary database. Then, should the primary database become unavailable for any reason, Autonomous Data Guard can convert the standby database to the primary database and, as such, it will begin servicing your applications.

The primary and standby databases are often called *peer* databases of each other. You can have up to two standby databases per Autonomous Container Database.

① Note

Applications must be configured to use Transparent Application Continuity (TAC) to gain the full benefit of the database availability features provided by Autonomous Data Guard.

The following diagram shows how each standby database is kept synchronized with the primary database.



Changes made to the primary database are recorded in the primary database's redo log. Autonomous Data Guard transmits these redo records as a stream over the network to the standby database's redo log. Then, the standby database applies these records to the standby database. In this way, the standby database is kept synchronized with the primary database.

Synchronization is nearly instantaneous, but, as the process just described implies, there are two operations that consume time: transporting the redo records to the standby database and applying the redo records to the standby database. The first of these is called the *transport lag*, and the other is called the *apply lag*. You can view current lag values for an Autonomous AI Database from the database's Details page from **Autonomous Data Guard** under **Resources**, under Resources in the side menu. You can view current lag values across all the Autonomous AI Databases in a container database from the container database's Details page in a similar fashion.

Note

With multiple standby databases, Cascaded Redo Transport is not supported.

Configuring Autonomous Data Guard

In Autonomous AI Database on Dedicated Exadata Infrastructure, you configure and manage Autonomous Data Guard at the Autonomous Container Database (ACD) level. You can enable Autonomous Data Guard for already provisioned ACDs and add up to two standby ACDs from its **Details** page using the Oracle Cloud Infrastructure console. See [Enable Autonomous Data Guard on an Autonomous Container Database and Add a Second Standby Autonomous Container Database](#) for instructions.

- You can now create and manage Autonomous Data Guard between an ACD in Oracle Cloud Infrastructure (OCI) and an ACD in Amazon Web Services (AWS).
- You can add a standby ACD from AWS region to the already provisioned ACD in the OCI region. Alternatively, you can add a standby ACD in the OCI region to an already provisioned ACD in the AWS region.

Please note the following before configuring Autonomous Data Guard:

- Autonomous AI Databases deployed on Exadata Cloud@Customer must have **Port 1522** open to allow TCP traffic between primary database and standby database in an Autonomous Data Guard setup.
- Autonomous Data Guard cannot be enabled on an ACD with an active maintenance run scheduled within the next three days. You can either run the active maintenance first and then enable Autonomous Data Guard or change the maintenance run schedule so that it does not begin until the second standby database is added.
- Adding a second standby database requires an automatic non-rolling restart for the the first standby database. The primary database is not affected by this non-rolling restart.

Configure Autonomous Data Guard with customer-managed keys

In Autonomous AI Database on Dedicated Exadata Infrastructure, you can configure and manage Autonomous Data Guard with customer-managed keys at the Autonomous Container Database (ACD) level. You can enable Autonomous Data Guard for already provisioned ACDs and add up to two standby ACDs from its Details page using the Oracle Cloud Infrastructure console. See [Enable Autonomous Data Guard on an Autonomous Container Database and Add a Second Standby Autonomous Container Database](#) for instructions.

Please note the following before configuring Autonomous Data Guard with customer-managed keys:

- If you are using Oracle Cloud Infrastructure Key Management System (OCI KMS) and want to enable Cross-Region Autonomous Data Guard:
 - You first need to replicate the OCI vault to the region where you want to add the standby database. See [Replicating Vaults and Keys](#) for more details.
 - You can only have the primary and standby databases in a maximum of 2 regions. That is, if you want to add a second standby and you have already used cross-region for the first standby, then the second standby needs to be in the primary region or the first standby region.

Note

Virtual vaults created before the cross-region vault replication feature was introduced can't be replicated across regions. Create a new vault and new keys if you have a vault that you need to replicate in another region and replication isn't supported for that vault. However, all private vaults support cross-region replication. See [Virtual vault cross-region replication](#) for details.

- If you are using Oracle Key Vault (OKV) and want to enable Cross-Region Autonomous Data Guard, ensure that you have added Connection IP addresses for your OKV Cluster in the Key Store.

Note

If you are configuring Autonomous Data Guard between an ACD in an OCI region and an ACD in AWS region, you can only use Oracle-managed Keys or Oracle Key Vault. You cannot use OCI KMS Keys or AWS KMS Keys.

Role Transitions and Operations

After an Autonomous Container Database (ACD) is created, you can change the role of the peer databases using either a switchover or a failover operation. If automatic failover is enabled, Autonomous Data Guard automatically performs a failover operation whenever the primary database become unavailable, for any reason.

A **switchover** is a role reversal between the primary database and its standby database. A switchover ensures no data loss. During a switchover, the primary database transitions to the standby role, and the standby database transitions to the primary role. To perform a switchover operation, see Switch Roles in an Autonomous Data Guard Configuration.

A **failover** is when the primary database is unavailable. The failover results in a transition of the standby database to the primary role. If automatic failover is not enabled, you can perform a manual failover as described in Fail Over to the Standby in an Autonomous Data Guard Configuration.

The availability and status of the database after a failover operation is characterized by two recovery objectives:

- **Recovery Time Objective (RTO).** The RTO the maximum amount of time required for the database to become available to applications after a failover, and is related to some degree to the apply lag at the time of the failure. For Autonomous Data Guard, the RTO is seconds up to two minutes.

- **Recovery Point Objective (RPO).** The RPO is the maximum duration of potential data loss from the failed primary database, and is related to some degree to the transport lag at the time of the failure. For Autonomous Data Guard, the RPO is near-zero.

After a failover, the failed primary becomes a **Disabled Standby** and remains unavailable for any database connection. You can re-enable it and turn it into a healthy standby by performing a **reinstat** operation. Once a failed primary has been reinstated as a standby, you can perform a switchover to return it to its original primary role. To perform a reinstat operation, see Reinstat the Disabled Standby in an Autonomous Data Guard Configuration.

Automatic Failover or Fast-Start Failover

With automatic failover, whenever the primary ACD becomes unavailable because of a region failure, an availability domain failure, a failure of the Exadata Infrastructure or Autonomous Exadata VM Cluster (AVMC), or the failure of the ACD itself, it automatically fails over to the standby ACD. This is also known as Fast-Start Failover.

You cannot enable automatic failover while configuring Autonomous Data Guard on an ACD. Automatic failover can only be enabled or disabled while updating the Autonomous Data Guard settings from the ACD **Details** page.

Note

Automatic failover cannot be enabled for Autonomous AI Databases deployed in Exadata Cloud@Customer with cross-region Autonomous Data Guard setup.

You can not add a second standby ACD with automatic failover enabled for the first standby ACD. So, disable automatic failover using [Update Autonomous Data Guard Settings](#) before the creating the second standby ACD, and re-enable it later, if needed.

Both the maximum performance and maximum availability protection modes support automatic failover:

- In the **Maximum availability** mode, automatic failover guarantees zero data loss.
- In the **Maximum performance** mode, automatic failover ensures that the standby database does not fall behind the primary database beyond the value specified for **Fast Start failover lag limit**. By default, **Fast Start failover lag limit** is set to 30 seconds and is applicable only to the Maximum performance mode. In this case, automatic failover is only possible when the apply lag (potential data loss) of the standby does not exceed the configured lag limit. You can modify the Fast Start failover lag limit to any value between 5 and 3600.

See [Update Autonomous Data Guard Settings](#) for more details.

Besides hardware failures, availability domain outages, and regional outages, there are a few more database health conditions that can trigger a Fast-Start Failover, as listed below:

Database Health Condition	Description
Corrupted Controlfile	Controlfile is permanently damaged because of a disk failure.
Corrupted Dictionary	Dictionary corruption of a critical database. Currently, this state can be detected only when the database is open.
Datafile Write Errors	Write errors are encountered in any data files, including temp files, system data files, and undo files.

As a result of automatic failover, the role of the failed primary database becomes Disabled Standby and, after a brief period, the standby database assumes the role of the primary database. After automatic failover concludes, a message is displayed on the details page of the disabled standby database advising you that failover has occurred.

After the service resolves the former primary Autonomous Container Database issues, you can perform a manual switchover to return both databases to their initial roles. Once you provision the standby database, you can perform various management tasks related to the standby database, including:

- Manually switching over a primary database to a standby database.
- Manually failing over a primary database to a standby database.
- Reinstating a primary database to standby role after failover.
- Terminating a standby database.

In an Autonomous Data Guard setup with multiple standby databases and automatic failover:

- Manual failovers require you to manually reinstate the original primary database, which becomes the new standby database.
- Whenever an automatic failover occurs, Autonomous AI Database on Dedicated Exadata Infrastructure tries to reinstate the old primary as a standby. However, if that attempt fails, then it must be manually reinstated.

Snapshot Standby Database

A snapshot standby database is a fully updateable standby database created by converting a standby Autonomous Container Database (ACD) to a snapshot standby ACD. See [Convert Physical Standby to Snapshot Standby](#) for step-by-step instructions.

A snapshot standby database receives and archives, but does not apply, redo data from the primary database. However, it increases your Recovery Time Objective (RTO) because real-time changes from the primary database are not applied.

Snapshot standby feature supports various use cases, but here are the primary use cases;

- Connect primary and standby application instances to primary and standby databases in read-write mode to perform initial configurations.
- Patch snapshot standby database first and test with your standby application instance to confirm patch stability. This requires converting the physical standby to a snapshot standby first, so the patch can be applied on the snapshot standby.

Note

You can not convert a physical standby Autonomous Container Database to a snapshot standby with automatic failover enabled.

While converting to a snapshot standby, you can either activate new database services that are active only in snapshot mode or use the same set of services used with the primary database. However, activating primary database services on the snapshot standby database may result in snapshot standby connection requests forwarded to the primary database or vice-versa if you use incorrect database connect strings. Hence, you must be careful to use appropriate connect string while connecting to your primary and snapshot standby database.

Note

When you create new services with snapshot standby, wallets for all the Autonomous AI Databases in the snapshot standby ACD are updated. To access the database, reload the wallets from standby Autonomous AI Databases and use snapshot standby connection strings.

You can convert the snapshot standby ACD back to a physical standby ACD from the Oracle Cloud Infrastructure (OCI) manually. See *Convert Snapshot Standby to Physical Standby* for detailed instructions. If a snapshot standby is not converted to a physical standby manually, it will be automatically converted back to a physical standby after 7 days from its creation. In any case, converting the snapshot standby back to a physical standby will discard all the local updates to your snapshot standby databases and apply the redo data received from the primary databases.

When a standby ACD is in the snapshot standby mode, you cannot perform the following operations on the primary ACD:

- Create or terminate Autonomous AI Databases
- Scale up or scale down Autonomous AI Databases
- Restore Autonomous AI Databases

If the situation demands, you can manually **failover** to a snapshot standby from the primary database. In that case, failover converts your snapshot standby database to a physical standby database by discarding all the local updates made to your snapshot standby and applying data from the primary database. See *Fail Over to the Standby in an Autonomous Data Guard Configuration* for step-by-step instructions.

A **switchover** between the primary database and its snapshot standby database is not allowed. You must manually convert your snapshot standby to a physical standby before attempting a switchover.

Accessing Standby Databases from Client Applications

In an Autonomous Data Guard configuration, your client applications normally connect to and perform operations on the primary database.

Connecting to the Physical Standby Database

In addition to this normal connectivity, Autonomous Data Guard provides you the option to connect client applications **that perform read-only operations** on the standby database. To take advantage of this option, client applications connect to the database using database service names that include "_RO" (for "read only"), as described in *Predefined Database Service Names for Autonomous AI Database*.

Connecting to the Snapshot Standby Database

Autonomous Data Guard also lets you connect client applications **that perform read-write operations** to the snapshot standby database. These operations are local to the snapshot standby database and do not modify its primary database. To connect to a snapshot standby database, client applications can use database service names that include "_SS" (for "snapshot standby"), as described in *Predefined Database Service Names for Autonomous AI Databases*.

Note

When the standby database is in the snapshot standby mode, all the database services that include "_RO" services in their name are inactive and can not be used for connections.

Monitoring Lag Times

As your databases that use Autonomous Data Guard are running, you can monitor transport lag and apply lag times from the database's (or container database's) Details page by choosing **Autonomous Data Guard Groups**. You can also use the OCI console or observability APIs to monitor transport lag and configure alarms and notifications. See Database Observability with Autonomous AI Database Metrics for more information.

You should expect to see minor fluctuations over time as the workload on your database ebbs and flows. However, if you notice a continuing upward trend in lag time, you can take these actions to resolve the situation:

- **Upward Trend in Apply Lag.** A continuing upward trend in apply lag indicates that the standby database doesn't have sufficient capacity to keep up with the redo records coming from the primary database. To resolve this situation, scale up the OCPUs of the database, as described in Add CPU or Storage Resources to a Dedicated Autonomous AI Database.
- **Upward Trend in Transport Lag.** A continuing upward trend in transport lag indicates a network performance issue. Oracle Cloud operations staff constantly monitors network performance, so you should see the situation resolve itself without you taking any action. However, if you want, you can bring the situation to the operations staff by raising a service request, as described in Create a Service Request in My Oracle Support.

Autonomous Data Guard Configuration Options

When you configure Autonomous Data Guard, you specify which Exadata Infrastructure and Autonomous Exadata VM Cluster resources you want the standby database created in, and you specify which data protection mode you want to use.

You have the following choices when specifying which Exadata Infrastructure and Autonomous Exadata VM Cluster resources to use for the standby:

- **In a different region** from the primary database's Exadata Infrastructure and Autonomous Exadata VM Cluster:
This choice provides the highest level of protection against disasters, including a catastrophic loss of external network connectivity or power to an entire region.

To make best use of this cross-region protection, your application tier also needs to be configured to support cross-region protection. Therefore, Oracle recommends that you choose this option if your application tier is already configured this way or if you are willing to reconfigure it to support cross-region protection.

If you choose to locate the standby database in a different region, Oracle recommends that you use the **Maximum Performance** protection mode.
- **In a different availability domain (AD)** from the primary database's Exadata Infrastructure and Autonomous Exadata VM Cluster:
This choice provides a high level of protection against disasters, including a catastrophic loss of external network connectivity or power to an availability domain within a region.

This choice provides a good balance between data protection and simplicity of configuration in your application tier.

If you choose to locate the standby database in a different availability domain, Oracle recommends that you use the **Maximum Availability** protection mode.

- **In the same availability domain (AD)** as the primary database's Exadata Infrastructure and Autonomous Exadata VM Cluster:
This choice provides a minimum level of protection against disasters, and Oracle recommends that you not choose it.

If the primary database's Exadata Infrastructure and Autonomous Exadata VM Cluster resources are in a region that has only one availability domain, Oracle recommends that you use the "in a different region" option.

If you do choose to locate the standby database in the same availability domain, Oracle recommends that you use the **Maximum Availability** protection mode.

- **In a different tenancy** from the primary database's Exadata Infrastructure and Autonomous Exadata VM Cluster:
APPLIES TO: 🟢 Oracle Public Cloud only

This choice allows you to add a standby database in a different tenancy from the primary database, letting your database failover or switchover to that cross-tenancy standby database. You can also create a snapshot standby in the remote tenancy. Having a cross-tenancy standby database can be helpful with database migration across tenancies.

Cross-tenancy standby databases:

- Can be enabled with either ECPU or OCPU compute model. The standby database must use the same compute model as the primary database.
- Supports automatic failover. However, Automatic failover cannot be enabled for Autonomous AI Databases deployed in Exadata Cloud@Customer with cross-region Autonomous Data Guard setup.
- Cannot be added using the Oracle Cloud Infrastructure console. You can only add a cross tenancy standby database using CLI or REST API. Once you add the standby database, you can view the cross-tenancy standby database, perform a failover or switchover to the cross-tenancy standby database from the Oracle Cloud Infrastructure console.

About Protection Modes

Autonomous Data Guard provides these data protection modes:

- **Maximum Availability.** This protection mode provides the highest level of protection that is possible without compromising the availability of the primary database.

The primary database does not commit transactions until it receives acknowledgment that the data has been received on the standby, (*not* that it has been written to disk). If the primary database does not receive this acknowledgment within 30 seconds, it operates as if it were in maximum performance mode to preserve primary database availability until it again receives acknowledgments in a timely fashion.

This protection mode ensures zero data loss except in the case of certain double faults, such as failure of a primary database after failure of the standby database.

- **Maximum Performance.** This is the default protection mode. It provides the highest level of data protection that is possible without affecting the performance of the primary database.

The primary database commits transactions as soon as all redo data generated by those transactions has been written to its online redo log. It also sends redo data to the standby database, but this is done asynchronously with respect to transaction commitment, so

primary database performance is unaffected by delays in writing redo data to the standby database.

This protection mode offers slightly less data protection than maximum availability mode and has minimal impact on primary database performance.

You can change the protection mode in an Autonomous Data Guard setup from the Oracle Cloud Infrastructure (OCI) console. See [Update Autonomous Data Guard Settings](#) for step-by-step instructions.

For more information about protection modes in Oracle Data Guard (which underlies the Autonomous Data Guard feature), see [Oracle Data Guard Protection Modes](#) in *Oracle Data Guard Concepts and Administration*.

Best Practices while Configuring Autonomous Data Guard

While Autonomous AI Database lets you create up to two standby ACDs with Autonomous Data Guard, you can choose to use single or multiple standby ACDs, depending on your requirement. However, to use the most resilient disaster recovery option that an Autonomous AI Database offers, you can add **one local standby** ACD and **one remote or cross-region standby** ACD with **maximum availability** as the data protection mode.

Let's understand the benefits of this design:

- **Local Standby:**
 - Automatic failover to a local standby in the same region provides significant local disaster isolation and application failover simplicity.
 - The business value of a local standby database is seen in zero data loss failover and application downtime reduced to seconds.
 - Applications automatically and transparently fail over to the local standby, maintaining the same latency between application servers and the database. This is particularly important for OLTP and package applications because higher latency can significantly impact throughput and overall application response time.
- **Remote Standby:**
 - If a regional disaster makes the primary and local standby systems inaccessible, the application and database can fail over to the remote standby.
 - Even though database downtime is still very low when a regional disaster occurs, the application downtime can be higher due to additional orchestration required for DNS, application, and database failover operations to the secondary region.
- **Maximum Availability:**
 - If automatic failover or fast start failover (FSFO) is enabled, whenever primary ACD becomes unavailable, Autonomous Data Guard fails over to the local standby with zero data loss and no change to database latency in application.
 - If automatic failover or fast start failover (FSFO) is enabled, whenever the entire primary region becomes inaccessible, system fails over to the remote standby with a potential data loss.

How Autonomous Data Guard Affects Standard Management Operations

In some cases, the standard management operations you perform on Autonomous Container Databases work differently on the primary and standby container databases in an Autonomous Data Guard configuration as compared to standard container databases. The following list describes these differences.

- **Change the maintenance schedule**
Maintenance scheduling of a primary container database and its standby are linked: maintenance on the standby is performed a number of days before maintenance on the primary. The default is 7 days; you can choose from 1 to 7 days when you create the primary container database or later by editing its Maintenance Details.
- **Change the maintenance type**
The maintenance type of a primary container database and its standby must be the same. You choose the maintenance type for both the primary and standby when you create the primary container database or later by editing its Maintenance Details.
- **Disable automatic backups**
You *can not* disable automatic backups while provisioning an Autonomous Container Database (ACD) with Autonomous Data Guard.
- **Manage scheduled maintenance**
You can manage scheduled maintenance of a primary container database and its standby separately. However, because maintenance of the two are linked, you must perform scheduled maintenance on the standby before the primary if you choose to override the scheduled maintenance time.
- **Move to a different compartment**
You can move primary and standby container databases to different compartments separately and independently, just as though they were standard container databases. However, as with standard container databases, you should exercise extreme caution when moving a container database to ensure that the container database remains accessible to the appropriate groups of cloud users.
- **Restart**
You can restart primary and standby container databases separately and independently, just as though they were standard container databases.
- **Rotate the encryption key**
You can rotate the encryption keys from the primary ACD or primary database.
- **Terminate**
You can terminate primary and standby container databases separately. However, the consequences of terminating a primary container database and terminating a standby container database differ:
 - Terminating a **primary container database** terminates both the primary and standby container databases. You cannot terminate a primary container database that contains Autonomous AI Databases.
 - Terminating a **standby container database** terminates the standby container database and removes it from the Autonomous Data Guard configuration. If it has only a primary remaining, the Autonomous Data Guard configuration is removed, turning the primary into a standalone container database.

Step-by-Step Guides

For step-by-step guidance on managing the Autonomous Data Guard configuration in an Autonomous Container Database, see:

- Enable Autonomous Data Guard on an Autonomous Container Database
- View the Status of an Autonomous Data Guard Configuration
- Switch Roles in an Autonomous Data Guard Configuration
- Fail Over to the Standby in an Autonomous Data Guard Configuration
- Reinstate the Disabled Standby in an Autonomous Data Guard Configuration

- Update Autonomous Data Guard Settings
- Convert Physical Standby to Snapshot Standby
- Convert Snapshot Standby to Physical Standby
- Add a Second Standby Autonomous Container Database

You can also use API to view and manage Autonomous Data Guard configuration. For more details, see [API to Manage Autonomous Data Guard Configuration](#).

Backup and Restore Autonomous AI Database on Dedicated Exadata Infrastructure

To support high availability, Autonomous AI Database on Dedicated Exadata Infrastructure automatically backs up your database for you. The retention period for backups is up to 95 days based on the Backup Retention Policy chosen for the Autonomous Container Database (ACD). You can restore and recover your database to any point-in-time in this retention period.

While backing up a database, the database is fully functional; however, during the backup, life cycle management operations are not allowed. For example, stopping the database is not allowed during the backup.

About Backup and Recovery

By default, Autonomous AI Database is configured to back up your database periodically. You can disable automatic backups while provisioning an ACD and enable them from the Oracle Cloud Infrastructure (OCI) console anytime later. Once the automatic backups are enabled you can not disable them for an ACD.

Autonomous AI Databases on Oracle Public Cloud are backed up to Object Storage with daily incremental backups and weekly full backups or Autonomous Recovery Service with daily virtual full backups, depending on their backup destination. Autonomous AI Database on Oracle Database@AWS are backed up to either OCI Object Storage or Amazon Simple Storage (S3) with daily incremental backups and weekly full backups or Autonomous Recovery Service with daily virtual full backups, depending on their backup destination.

With automatic backups enabled, you have an option to take any of the following additional backups:

- **Cross region backups**, also known as remote backups, are backups stored in a different region than the source database. This backup option is only available for ACDs deployed on Oracle Public Cloud. In case of a region failure, you can use this backup to recover your database.

Cross region backups:

- Are additional backups on top of the default automatic backups. They are like a second backup taken in a region that you've selected.
- Are supported only at the ACD level and include all Autonomous AI Databases within the ACD, as well as all archive logs.
- Are exclusively intended for cloning into a new ACD and cannot be used for in-place restore operations.
- Can be cloned into a different Autonomous Exadata VM Cluster (AVMC). See [Clone an Autonomous Container Database from Backup](#) for instructions.

Note

Cross-region backups can only be enabled on ACDs configured without Autonomous Data Guard.

- **Long-term backups** are the backups that are retained beyond the backup retention policy (period) defined at the ACD level. When creating a long-term backup, you can specify its retention period, ranging from 90 days to 10 years.

Long-term backups:

- Are exclusively intended for cloning into new Autonomous AI Databases and cannot be used for Autonomous AI Database in-place restore operations.
- Are supported only at the Autonomous AI Database level.
- Cannot be taken on Autonomous AI Databases that reside in an ACD that has not enabled backups.
- Are listed on the Autonomous AI Database details page in the section on Backups. They are also listed under Autonomous standalone backups on the OCI console.

- **Manual Backups** are on-demand backups taken manually from the OCI console. For example, before making a significant update to your application, you can take a manual backup as an option to restore and recover later.

Manual backups:

- Are cumulative incremental backups containing all the incremental backups from the last full backup if the backup destination is Object Storage.
- Are daily virtual full backups if the backup destination is Autonomous Recovery Service.
- Are retained for up to 95 days, depending on the backup retention policy (period) chosen for the ACD.
- Are supported only at the Autonomous AI Database level.

Backup Retention Policy (Period)

You can define the **backup retention policy/period** while provisioning an ACD or modify it later from its details page on the Oracle Cloud Infrastructure console.

The backup retention policy value defaults to *15 days* and can be changed to any value between *7 to 95 days*.

On Exadata Cloud@Customer deployments:

- You can not modify the backup retention policy if you use **Recovery Appliance** or **Local Storage** as your ACD's backup destination. In the case of **Recovery Appliance**, this value is controlled by the Recovery Appliance protection policy.
- For **Object Storage** and **Network File System (NFS)** backup destination types, the backup retention policy defaults to *30 days* and can be changed to any value between *7 to 95 days*.

Retention Lock:

You can retain the automatic backups even after any (or all) of the Autonomous AI Database components, like Exadata Infrastructure, Autonomous VM Cluster (AVMC), ACD, and Autonomous AI Database are terminated. The timeline to retain backups after an ACD is terminated is applicable only for automatic backups and not for long-term backups. You can retain the backups of a terminated ACD for either up to 72 hours or until the end of its backup

retention period. Additionally, on ACD termination, when you choose to retain backups for the duration of the backup retention period, you can enable Retention Lock. Enabling a backup retention lock disallows deleting all the database backups, including the long-term backups, before their retention period completes. You can not delete these backups until the retention period expires.

You can enable the retention lock either while provisioning the ACD or later. Once enabled, you can disable the retention lock within 14 days, beyond which the setting becomes permanent. If you have enabled retention lock for your ACD, then the retention lock applies to both automatic backups and long-term backups of your Autonomous AI Databases.

If you have enabled retention lock:

- After the ACD is terminated, all automatic backups and long-term backups are retained for their retention period.
- The retention period can only be increased later and cannot be decreased.
- At the end of the backup retention period of the ACD, all automatic backups are deleted. However, if you have configured NFS for storing the automatic backups, then you have to manually delete the backups. Automatic backups are guaranteed to remain till the end of the backup retention period.
- After the ACD is terminated, long-term backups cannot be deleted and continue to remain until they expire. All long-term backups are automatically deleted after they expire. However, if you have configured NFS for storing the long-term backups, then you need to manually delete them after they expire.

If you have not enabled retention lock:

- After the ACD is terminated, all automatic backups are deleted based on the retention period that is set, which is either 72 hours or the backup retention period of the ACD. However, if you have configured NFS for storing the automatic backups, then you have to manually delete the automatic backups.
- The retention period can be increased or decreased at any time.
- The long-term backups can be deleted at any time. You can also choose to delete them when the Autonomous AI Database gets terminated. However, if you have configured NFS for storing the long-term backup and the corresponding ACD is terminated, then you have to manually delete the backups after they expire.

Backup Destination

You can associate a backup destination for the backups of Autonomous AI Databases created in an ACD.

On Oracle Public Cloud deployments, you can choose either Autonomous Recovery Service or Object Storage as the backup destination. The default is Object Storage and the recommended option is Autonomous Recovery Service. For Autonomous AI Database on Oracle Database@AWS, you can choose either Autonomous Recovery Service, OCI Object Storage or Amazon Simple Storage (AWS S3). The default is AWS S3 and the recommended option is Autonomous Recovery Service.

Below listed are the backup destination options for Autonomous AI Database deployments.

Table 3-1 Backup destination options

Backup destination type	Supported Deployments	Notes
Object Storage	Autonomous AI Database on Oracle Public Cloud Autonomous AI Database on Exadata Cloud@Customer Autonomous AI Database on Oracle Database@AWS	Stores weekly full and daily incremental backups in an Oracle-managed object storage container on Oracle Cloud Infrastructure. For Exadata Cloud@Customer deployments, if you choose Object Storage as the type, you can optionally specify an internet HTTP proxy to use when connecting to the storage container. Oracle recommends using a proxy when possible for enhanced security.
Autonomous Recovery Service (ARS)	Autonomous AI Database on Oracle Public Cloud Autonomous AI Database on Oracle Database@AWS	Stores daily virtual full backups. This is the recommended option.
Network File System (NFS)	Autonomous AI Database on Exadata Cloud@Customer	Stores backups in a Network File System (NFS) storage location. If you choose Network File System (NFS) as the type, select an already defined backup destination that uses Network File System (NFS) storage. See Prerequisites for Backup Destinations for Exadata Cloud@Customer before configuring backup destinations on an NFS storage for Exadata Cloud@Customer If you choose NFS as a backup destination type, then you have to manually delete the backups after the backup retention period if the corresponding ACD is terminated. See See Backup Retention Policy for more details.

Table 3-1 (Cont.) Backup destination options

Backup destination type	Supported Deployments	Notes
Recovery Appliance	Autonomous AI Database on Exadata Cloud@Customer	Stores backups in one of your previously defined backup destinations that uses Oracle Zero Data Loss Recovery Appliance. If you choose Recovery Appliance as the type, select a previously defined backup destination that uses Oracle Zero Data Loss Recovery Appliance, the DB_UNIQUE_NAME of the ACD, and the VPC user name password. For a Zero Data Loss Recovery Appliance backup destination: - The appliance must be configured with a virtual private catalog (VPC) user, which is used for taking the backups. - The appliance must be configured with the unique database name of the database being backed up, and a mapping to the VPC user. For example: <pre>EXEC dbms_ra.add_db(db_uni que_name=>'<ACD db unique name>', protection_policy_nam e=>'VF_7DAYS', reserved_space=>'500G ');</pre> - The appliance must be configured with a mapping to the VPC user. For example: <pre>EXEC dbms_ra.grant_db_acce ss(db_unique_name=>'< ACD db unique name>', username=>'<VPC user>');</pre> - The appliance must be accessible from the Oracle Exadata Database Service on Cloud@Customer system using the Oracle Net Services connection string, which is provided by the Zero

Table 3-1 (Cont.) Backup destination options

Backup destination type	Supported Deployments	Notes
		Data Loss Recovery Appliance administrator. • All primary and standby ACDs in the same Autonomous Data Guard cannot share the same ZDLRA or any other Recovery Appliance instance. • ZDLRA instances must not have replication enabled. • The following example query can be used to test your ZDLRA configuration. This query should return the value "1". <pre>select count(1) from rasys.db where REG_DB_UNIQUE_NAME = '<ACD db unique name>';</pre>

**i Not
e**

Make sure to provide the connection string that connects to the recovery appliance in an Oracle "easy connect" string format,

Table 3-1 (Cont.) Backup destination options

Backup destination type	Supported Deployments	Notes
		that is, <host>:<port>/<service name>, where <host> is the SCAN hostname of the Zero Data Loss Recovery Appliance. See Prerequisites for Backup Destinations for Exadata Cloud@Customer before configuring backup destinations on a Zero Data Loss Recovery Appliance location for Exadata Cloud@Customer See Configuring Recovery Appliance for Protected Database Access in the Zero Data Loss Recovery Appliance Administrator's Guide for additional information.
Local	Autonomous AI Database on Exadata Cloud@Customer	Stores backups on the local Exadata storage. This option is only available if you have configured Local storage while creating the Autonomous Exadata VM Cluster. Once an ACD is created with a backup destination type anything other than Local, it cannot be changed to Local backups later.
AWS S3	Autonomous AI Database on Oracle Database@AWS	Stores backups on AWS Object Storage.

Changing Backup Destinations

APPLIES TO:  Exadata Cloud@Customer only

You can change the backup destination type for an ACD with automatic backup enabled. At any given point in time, there can only be a maximum of two backup destinations associated with an ACD.

Table 3-2 Changing Backup Destinations

Current backup destination type	New backup destination types that are supported	Notes
Local	-	You cannot change the backup destination type if the current type is local.
Network File System (NFS)	NFS RA	-
Object Storage	RA	-
Recovery Appliance (RA)	RA NFS Object Storage	Changing the backup destination type from RA to Object Storage is only supported if the ACD was originally configured with Object Storage backups first.

Restore and Recover

You can restore the database using one of the following options:

- Use a specific backup for restoring the database
- Use a timestamp for point in time recovery from the OCI console
- Use a specific System Change Number (SCN) for restoring the database from the OCI console

The database goes into the unavailable state while restore is in progress. You cannot connect to the database in this state. The only life cycle management operation supported in unavailable state is terminate. When the restore operation finishes, your database is opened in the same state as before restoration. For all three types of restore (by SCN, by timestamp or by using specific backup) an SCN will be returned when the restore is completed.

When your database is restored, the value of the ADMIN user password is also restored. Therefore, it may have an old value that you no longer remember. You can set the password to a new value as described in [Unlock or Change the ADMIN Database User Password](#).

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.

In an Autonomous Data Guard setup:

- You can only restore a backup from the *primary* database.
- Database restore is not allowed if the standby database is in snapshot standby role. You must convert the standby Autonomous Container Database (ACD) to physical standby role

to restore this database. See Convert Snapshot Standby to Physical Standby for instructions.

Backup and Restore Operations

The following table lists the operations that you can perform on backups of Autonomous AI Database.

Operation	Details	Further Reference
Create a manual backup	You can create a manual (on-demand) backup from the Details page of an ACD on the OCI console.	Backup Your Autonomous AI Database Manually
Create a long-term backup	You can create a long-term backup from the Details page of an ACD on the OCI console.	Create a Long-Term Backup
Clone from a backup	You can create a point-in-time copy of a backup set of your Autonomous AI Database. You can clone from an automatic backup, manual backup (chosen from the list of backups or based on a timestamp), or a long-term backup.	For complete details about cloning from a backup, its requirements and limitations, refer to About Cloning Autonomous AI Database on Dedicated Exadata Infrastructure For instructions on cloning a manual or long-term backup from the OCI console, see Clone Autonomous AI Database from Backup. For instructions on cloning from an ACD backup, see Clone Autonomous Container Database from Backup.

Operation	Details	Further Reference
Edit backup settings	You can enable automatic backups (if disabled while provisioning the ACD) and modify the following backup settings from the Details page of an ACD from the OCI console: - Backup retention period to any value between 7 to 95 days. - Backup destination type, if deployed on Exadata Cloud@Customer. - HTTP Proxy. This is applicable only when Object Storage is selected as the backup destination for Exadata Cloud@Customer deployments. - Cross region backup copy, if deployed on Oracle Public Cloud. - Retain an automatic backup for either the duration of the backup retention period or for 72 hours after the ACD is terminated. - Enable Retention Lock. This is applicable only if you choose to retain backups for the duration of the backup retention period upon ACD termination.	For complete details about editing the backup settings, see Edit Autonomous Container Database Backup Settings.
Update a long-term backup	You can edit the retention period of a long-term backup or even delete a long-term backup from the Details page of an ACD on the OCI console.	Manage Long-Term Backups
Restore and recover your Autonomous AI Database	You can either restore the database from a specific backup or use a timestamp for point-in-time recovery from the Details page of the ACD on the OCI console.	Restore and Recover Your Autonomous AI Database
View Remote Backups	You can view a list of remote backups in a region from the OCI console.	View a List of Remote Backups
Clone an ACD from Backup	You can recover your database by cloning an ACD from a remote backup into an AVMC in a different region than the clone source.	Clone an Autonomous Container Database from Backup

You can also use API to perform backup and restore operations. For more details, see API Reference for Autonomous AI Database on Dedicated Exadata Infrastructure.

Backup and Restore Events

Oracle Database resources emit events, which are structured messages that indicate changes in resources. There are multiple events associated with database backup operations. You can find them in Events for Autonomous AI Database on Dedicated Exadata Infrastructure.

Best Practices for estimating Backup space

You need accurate estimation of backup storage requirements when deploying Autonomous AI Database on Dedicated Exadata Infrastructure, both in Oracle Public Cloud and Exadata Cloud@Customer environments. This section outlines the key factors and calculations necessary to ensure compliance with recovery policies and to avoid unanticipated storage utilization or costs.

Backup Management in Autonomous AI Database

Backups are managed automatically, with the following default schedule:

- Full Backup: Performed weekly.
- Incremental Backups: Performed daily, except on the day of the full backup.
- Archived Redo Logs: Generated daily and included in their own backup cycles.

Collect Accurate Source Metrics

For the most precise estimate, perform the base calculations on metrics from your existing Oracle AI Database environment, including:

- Total Container Database (CD) Size
- Average Daily Data Change Rate
- Daily ArchiveLog Generation Rate
- Actual Observed Compression Rates (especially if using Hybrid Columnar Compression)

Note

Oracle AI Database backups typically achieve a 50% compression rate, unless data is already compressed by Hybrid Columnar Compression (HCC), in which case little to no further compression should be assumed. Archive logs generally compress minimally.

Retention Planning

To maintain a given recovery window (e.g., 14 days), you must retain the latest full backup and all associated incremental and archived redo logs until they are all superseded by a new full backup and its incremental. Because of dependency between backups, for a 2-week retention policy, plan for up to 3 weeks of backup storage to account for overlap and to ensure your recovery window is always fully protected.

Additional Backup Scenarios

Manual Backups:

- Manual backups are cumulative incremental backups.
- The size typically equals the sum of all changes since the last full backup.

Long-Term Backups:

- Performed at the ADB level within the ACD.
- Each long-term backup's size is based on the individual ADB, not the entire ACD.
- Storage should be sized based on the number of long-term backups needed for every ADB and sum of all ADBs requiring long-term backup retention.

Special Note for Exadata Cloud@Customer with ZDLRA:

For environments leveraging the Zero Data Loss Recovery Appliance (ZDLRA):

- Only the initial full backup is physically stored.
- All further backups are incremental; virtual full backups are synthesized when needed.
- ZDLRA manages archivelog backups and retention, typically reducing storage overlap.
- Storage is required for a single full backup, cumulative incremental, and archived redo logs, further optimized by ZDLRA's internal deduplication mechanisms.

Note

For ZDLRA deployments, always refer to actual database metrics and consult ZDLRA documentation for precise sizing.

Example Calculation

Consider an environment with:

- ACD Size: 10 TB
- Daily Data Change Rate: 10% (1 TB)
- Archive Log Generation: 1 TB per day

Weekly Backup Profile:

- Full backup: 10 TB (before compression)
- Incrementals: $6 \times 1 \text{ TB} = 6 \text{ TB}$ (before compression)
- Archive logs: $7 \times 1 \text{ TB} = 7 \text{ TB}$
- After Typical Compression (50% for backups):
- Full + Incremental = $(10 \text{ TB} + 6 \text{ TB}) \times 50\% = 8 \text{ TB}$
- Archive logs: 7 TB (assume little compression)
- Total backups per week: 8 TB (compressed backups) + 7 TB (archive logs) = 15 TB
- Retention Sizing: 3-week storage requirement for 2-week retention policy: $15 \text{ TB} \times 3 = 45 \text{ TB}$ (plus minor additional overhead for metadata and operational variation)

Summary

- Always use recent and relevant operational metrics from your production workloads for estimation.
- Re-evaluate backup and retention policies when database characteristics or business requirements change.
- For environments with complex storage or compliance needs (such as ZDLRA integration), seek additional guidance from Oracle technical resources.

Table 3-3 Key Parameters for Estimating Backup Storage

Parameter	Typical Value / Recommendation
Full Backup Frequency	Weekly
Incremental Backup Frequency	Daily (non-full backup days)
Archivelog Generation Rate	Daily (estimate from observed rate)
Compression Rate	50% (except for HCC-compressed data)
Retention Policy	2 weeks (plan for 3 weeks of concurrent storage)
Storage Calculation	(Full + Incremental) × Compression + Archive Logs

Use OCI Full Stack Disaster Recovery on Autonomous AI Database on Dedicated Exadata Infrastructure

Full Stack Disaster Recovery is an Oracle Cloud Infrastructure (OCI) disaster recovery orchestration and management service that provides comprehensive disaster recovery capabilities for all layers of an application stack, including infrastructure, middleware, database, and application.

When you enable the OCI Full Stack Disaster Recovery (Full Stack Disaster Recovery) you can use Full Stack Disaster Recovery to perform switchover/failover operations or optionally perform database only Autonomous AI Database switchover/failover operations.

See [About Full Stack Disaster Recovery](#) for more information.

Prerequisite Steps for Autonomous Container Database with Full Stack Disaster Recovery

See Policies for Full Stack Disaster Recovery and Policies for Oracle Autonomous AI Database for the required policies and their usage with Full Stack Disaster Recovery.

If the disaster recovery topology contains any database related to the Oracle Autonomous Container Database, perform the following steps to prepare the databases for Full Stack Disaster Recovery:

1. Configure an Autonomous Data Guard association for the primary database. See Manage Autonomous Data Guard Configuration for instructions.
2. Add the Autonomous Data Guard peer (standby database) in the standby region that you intend to use for disaster recovery.

Enable Full Stack Disaster Recovery

Enable Full Stack Disaster Recovery by adding an Autonomous Container Database instance to a Disaster Recovery Protection Group.

For instructions, see [Add an Autonomous Container Database to a Disaster Recovery Protection Group](#).

After you create a Disaster Recovery Protection Group that includes an Autonomous Container Database, the Oracle Cloud Infrastructure console shows that Full Stack Disaster Recovery is enabled.

Use Full Stack Disaster Recovery to Perform Switchover or Failover Operations

After you enable Full Stack Disaster Recovery, when you perform a switchover or failover you have the option to use Full Stack Disaster Recovery.

For a switchover or failover you must initiate the operation from a Standby database. You have several options to access a Standby database:

- Select the region in Oracle Cloud Infrastructure Console and then access the Standby database.
- On the Primary database you can access a standby database from the Oracle Cloud Infrastructure Console. To access a standby database, from the Autonomous AI Database details page select **Disaster recovery** under **Resources** and click a link for a standby database in the **Peer Autonomous AI Database** column.

To perform a switchover or a failover:

1. Access the standby database and from the Autonomous AI Database details page. Under **Disaster recovery** click **Switchover** (or **Failover** when the Primary is not available) in the **Role** field.
As an alternative, to initiate a switchover you can select **More actions** and **Switchover** (or **Failover** when the Primary is not available).

This shows the Confirm switchover to peer dialog.

2. Select an option and confirm the operation.
 - a. Select **Switchover stack with Full Stack Disaster Recovery**.
 - b. In the **Select Disaster Recovery Protection Group** field, select a value.
 - c. Click **Goto DR protection group**.

This brings you to the Disaster Recovery DR protection group details page. From this page you can execute Full Stack Disaster recovery plans for the entire application stack that includes this Autonomous AI Database. See [Manage Disaster Recovery Plan Executions](#) for more information.

High Performance

High Performance Features in Autonomous AI Database on Dedicated Exadata Infrastructure

This article describes the key performance tuning features in Autonomous AI Database on Dedicated Exadata Infrastructure.

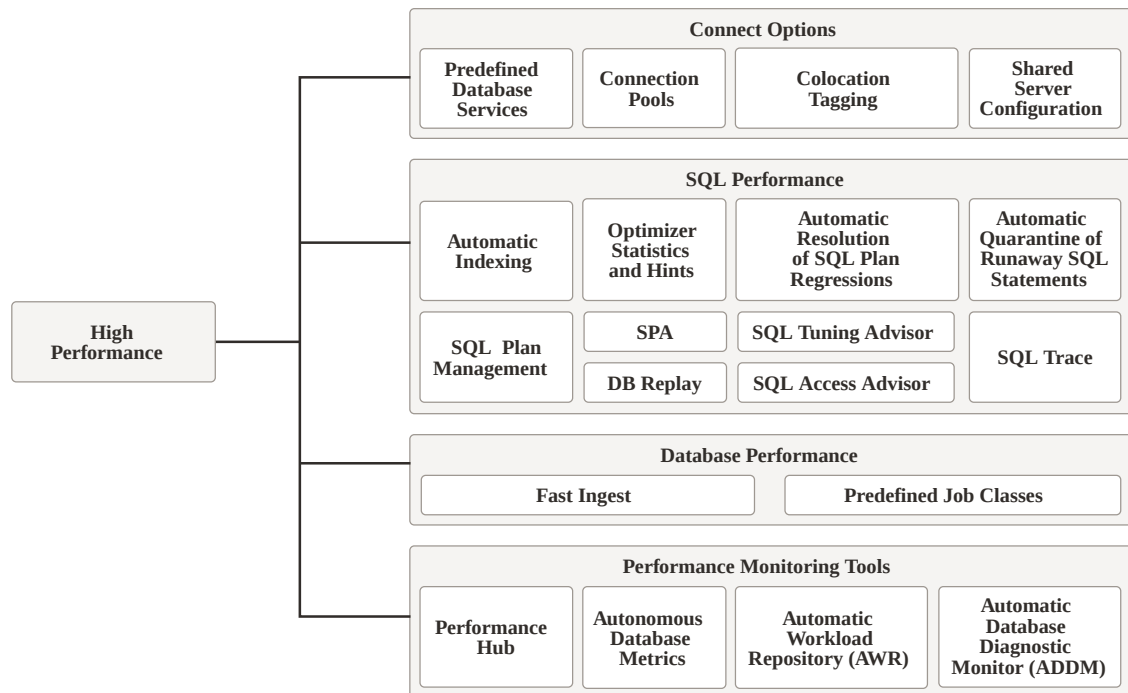
Note that throughout this section the term "you" is broadly used to mean any user in your organization who has the responsibility for performing certain tasks. In some cases, that's the application DBA, in others it's the application developer.

Autonomous AI Database includes several features that automatically monitor, analyze and optimize the performance of your database. For a complete list of the SQL tuning and performance management features of Oracle Autonomous AI Database, and instructions on how to use them, see Oracle Database 19c SQL Tuning Guide or Oracle Database 26ai SQL Tuning Guide.

You can see a broad categorization for the key performance tuning features of Autonomous AI Database depicted below.

✓ **Tip**

In the following image, you can click the feature you want to explore further.



Predefined Database Services

How your application connects to your database and how you code SQL calls to the database determine the overall performance of your application's transaction processing and reporting operations.

When making connections to your Autonomous AI Database, the performance of your application's interaction with the database depends on which database service you connect to. Autonomous AI Database provides multiple sets of database services to use when connecting to your database. These connection services are designed to support different kinds of database operations as described in *Predefined Database Service Names for Autonomous AI Databases*.

✓ **Tip**

Ensure to review the key characteristics of the predefined database services and the table that compares the different sets of database services based on these characteristics to decide which database service is more appropriate for your application's performance requirements.

Connection Pools

When making connections to your Autonomous AI Database, you can use connection pools to reduce the performance overhead of repeatedly creating and destroying individual connections. This is another factor that has great impact on the performance of your application's interaction with the database.

Quite often, the use of connection pools is considered only when designing or enhancing an application to provide continuous availability. However, the use of connection pools instead of individual connections can benefit almost every transaction processing application. A connection pool provides the following benefits:

- Reduces the number of times new connection objects are created.
- Promotes connection object reuse.
- Quickens the process of getting a connection.
- Controls the amount of resources spent on maintaining connections.
- Reduces the amount of coding effort required to manually manage connection objects.

Special-Purpose Connection Features

Oracle Net Services (previous called SQL*Net) provides a variety of connection features that improve performance in specific connection scenarios. These features are described in *Oracle Database 19c Net Services Administrator's Guide* or *Oracle Database 26ai Net Services Administrator's Guide*.

- **Colocation tagging** is one such feature that is useful in certain transaction processing applications. If your application repeatedly makes connections to the same database service, colocation tagging permits all such connections to be directed to the same database instance, bypassing the load-balancing processing normally done on the database side of connections. For more information, see *COLOCATION_TAG of Client Connections* in *Oracle Database 19c* or *Oracle Database 26ai*.
- **Shared Server Configuration** is another feature Autonomous AI Database supports for maintaining legacy applications designed without connection pooling. The shared server architecture enables the database server to allow many client processes to share very few server processes. This increases the number of users that the application can support. Using the shared server architecture for such legacy applications enables them to scale up without making any changes to the application itself.

You can enable shared server connections while provisioning an Autonomous Container Database (ACD) and this setting applies to all the databases created in it. See *Create an Autonomous Container Database* for instructions.

See also *Oracle Database 19c Net Services Administrator's Guide* or *Oracle Database 26ai Net Services Administrator's Guide* for more detailed information about shared servers, including features such as session multiplexing.

Once the Shared Server connection is enabled for your Autonomous Container Database, changing the connect string is not necessary. The default configuration is set to Dedicated.

Note

You can not disable a Shared Server for a specific Autonomous AI Database created under a Shared Server-enabled Autonomous Container Database, and you can not use a Dedicated connection for Autonomous AI Databases created under a Shared Server-enabled Autonomous Container Database.

SQL Performance Tuning Features

Great applications begin with well written SQL. Oracle Autonomous AI Database provides numerous features that enable you to build high performance applications and validate your SQL and PL/SQL code. Some of these features are listed below:

- Automatic Indexing
- Optimizer Statistics and Hints
- Automatic resolution of SQL plan regressions
- Automatic quarantine of runaway SQL statements
- SQL Plan Management
- SQL Tuning sets
- SQL Trace

As you develop your application, you can quickly learn how these features are affecting the SQL code you write and so improve your code by using the SQL Worksheet provided by both Oracle Database Actions (which is built into your Autonomous AI Database) and Oracle SQL Developer (a free application you install on your development system).

SQL Tracing

When an application operation takes longer than expected, getting a trace of all the SQL statements executed as part of this operation with details such as time spent by that SQL statement in the parse, execution, and fetch phases will help you identify and resolve the cause of the performance issue. You can use SQL tracing on an Autonomous AI Database to achieve this.

SQL tracing is disabled by default in Autonomous AI Database. You must enable it to start collecting the SQL tracing data. Refer to [Use SQL Tracing on Autonomous AI Database](#) for detailed instructions to enable and use SQL Tracing.

Optimizer Statistics

Autonomous AI Database gathers optimizer statistics automatically so that you do not need to perform this task manually and this helps to ensure your statistics are current. Automatic statistics gathering is enabled in Autonomous AI Database and runs in a standard maintenance window.

Note

For more information on maintenance window times and automatic optimizer statistics collection, see *Oracle Database 19c Administrator's Guide* or *Oracle Database 26ai Administrator's Guide* .

For more information on optimizer statistics see *Oracle Database 19c SQL Tuning Guide* or *Oracle Database 26ai SQL Tuning Guide* .

Optimizer Hints

Optimizer hints are special comments in a SQL statement that pass instructions to the optimizer. The optimizer uses hints to choose an execution plan for the statement unless prevented by some condition.

Autonomous AI Database defaults for the optimizer and PARALLEL hints differ depending on your workload:

- **Autonomous AI Lakehouse:** Autonomous AI Database with Data Warehouse *ignores* optimizer hints and PARALLEL hints in SQL statements *by default*.

If your application relies on hints, you can enable optimizer hints by setting the parameter `OPTIMIZER_IGNORE_HINTS` to `FALSE` at the session or system level using `ALTER SESSION` or `ALTER SYSTEM`.

For example, the following command enables hints in your session:

```
ALTER SESSION
SET OPTIMIZER_IGNORE_HINTS=FALSE;
```

You can also enable PARALLEL hints in your SQL statements by setting `OPTIMIZER_IGNORE_PARALLEL_HINTS` to `FALSE` at the session or system level using `ALTER SESSION` or `ALTER SYSTEM`.

For example, the following command enables PARALLEL hints in your session:

```
ALTER SESSION
SET OPTIMIZER_IGNORE_PARALLEL_HINTS=FALSE;
```

- **Autonomous AI Transaction Processing:** Autonomous AI Database *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`.

For example, the following command enables PARALLEL hints in your session:

```
ALTER SESSION
SET OPTIMIZER_IGNORE_PARALLEL_HINTS=TRUE;
```

Automatic Indexing

Automatic indexing automates the index management tasks in Autonomous AI Database. Automatic indexing is disabled by default in Autonomous AI Database.

Creating indexes manually requires deep knowledge of the data model, application, and data distribution. In the past, DBAs were responsible for making choices about which indexes to create, and then sometimes the DBAs did not revise their choices or maintain indexes as the conditions changed. As a result, opportunities for improvement were lost, and use of unnecessary indexes could become a performance liability.

The automatic indexing feature in Autonomous AI Database monitors the application workload and creates and maintains indexes automatically.

✓ Tip

For a "try it out" alternative that demonstrates these instructions, run *Lab 14: Automatic Indexing* in the [Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop](#).

Enable Automatic Indexing

- Use the DBMS_AUTO_INDEX.CONFIGURE procedure to enable automatic indexing. For example, executing the below statement enables automatic indexing in a database and creates any new auto indexes as visible indexes, so that they can be used in SQL statements.

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE', 'IMPLEMENT');
```

- Use the DBMS_AUTO_INDEX package to report on the automatic task and to set automatic indexing preferences.

📘 Note

Note: When automatic indexing is enabled, index compression for auto indexes is enabled by default.

Disable Automatic Indexing

Use the DBMS_AUTO_INDEX.CONFIGURE procedure to disable automatic indexing. For example, executing the below statement disables automatic indexing in a database so that no new auto indexes are created. However, the existing auto indexes remain enabled.

```
EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE', 'OFF');
```

For more information see *Managing Auto Indexes* in Oracle Database 19c Administrator's Guide or Oracle Database 26ai Administrator's Guide .

Fast Ingest

Fast ingest optimizes the processing of high-frequency, single-row data inserts into a database. Fast ingest uses the large pool for buffering the inserts before writing them to disk, so as to improve data insert performance.

The intent of fast-ingest is to support applications that generate lots of informational data that has important value in the aggregate but that doesn't necessarily require full ACID guarantees. Many applications in the Internet of Things (IoT) have a rapid "fire and forget" type workload, such as sensor data, smart meter data or even traffic cameras. For these applications, data might be collected and written to the database in high volumes for later analysis.

Fast ingest is very different from normal Oracle AI Database transaction processing where data is logged and never lost once "written" to the database (that is, committed). In order to achieve the maximum ingest throughput, the normal Oracle transaction mechanisms are bypassed, and it is the responsibility of the application to check to see that all data was indeed written to the database. Special APIs have been added that can be called to check if the data has been written to the database.

For an overview of fast ingest and the steps involved in using this feature, refer to *Using Fast Ingest* in Oracle Database 19c Performance Tuning Guide or Oracle Database 26ai Performance Tuning Guide .

To use fast ingest with your Autonomous AI Database, you must:

- **Enable the Optimizer to Use Hints:** Set the `optimizer_ignore_hints` parameter to FALSE at the session or system level, as appropriate.

Depending on your Autonomous AI Database workload type, by default `optimizer_ignore_hints` may be set to FALSE at the system level. See [Optimizer Statistics](#) for more information.

- **Create a Table for Fast Ingest:** Refer to Oracle Database 19c Performance Tuning Guide or Oracle Database 26ai Performance Tuning Guide for the limitations for tables to be eligible for Fast Ingest (tables with the specified characteristics cannot use fast ingest).

Predefined Job Classes with Oracle Scheduler

Autonomous AI Database includes predefined `job_class` values to use with Oracle Scheduler. These job classes let you group jobs that share common characteristics and behavior into larger entities so that you can prioritize among these classes by controlling the resources allocated to each class.

With predefined job classes, you can ensure that your critical jobs have priority and enough resources to complete. For example, for a critical project to load a data warehouse, you can combine all the data warehousing jobs into one class and prioritize it over other jobs by allocating a high percentage of the available resources. You can also assign relative priorities to the jobs within a job class.

The predefined `job_class` values, TPURGENT, TP, HIGH, MEDIUM and LOW map to the corresponding consumer groups. These job classes allow you to specify the consumer group a job runs in with `DBMS_SCHEDULER.CREATE_JOB`.

The `DBMS_SCHEDULER.CREATE_JOB` procedure supports `PLSQL_BLOCK` and `STORED_PROCEDURE` job types for the `job_type` parameter in Autonomous AI Database.

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

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name => 'update_sales',
    job_type => 'STORED_PROCEDURE',
    job_action => 'OPS.SALES_PKG.UPDATE_SALES_SUMMARY',
    start_date => '28-APR-19 07.00.00 PM Australia/Sydney',
    repeat_interval => 'FREQ=DAILY;INTERVAL=2',
    end_date => '20-NOV-19 07.00.00 PM Australia/Sydney',
    auto_drop => FALSE,
    job_class => 'HIGH',
    comments => 'My new job');
END;
/
```

Notes for Oracle Scheduler:

- To use `DBMS_SCHEDULER.CREATE_JOB` additional grants for specific roles or privileges might be required. The `ADMIN` user and users with `DWROLE` have the required `CREATE SESSION` and `CREATE JOB` privileges. If a user does not have `DWROLE` then grants are required for `CREATE SESSION` and `CREATE JOB` privileges.
- The `instance_id` job attribute is ignored for Oracle Scheduler jobs running on Autonomous AI Database.

See *Scheduling Jobs with Oracle Scheduler* in Oracle Database 19c or Oracle Database 26ai for more information on Oracle Scheduler and `DBMS_SCHEDULER.CREATE_JOB`.

See *SET_ATTRIBUTE Procedure* in Oracle Database 19c or Oracle Database 26ai for information on job attributes.

Performance Monitoring and Tuning Tools

Several situations such as changing workloads, resource limitations on application and database servers, or simply network bottlenecks can give rise to application performance issues. Oracle provides a wide range of tools to help you monitor performance, diagnose performance issues, and tune your application or the database to resolve the issue. Some of them are listed below:

Tool	Details
Performance Hub	A readily available feature-rich tool that is available in the Oracle Cloud Infrastructure (OCI) console. Performance Hub also comes in-built with Oracle Database Actions and Oracle Enterprise Manager. See <i>Monitor Database Performance with Performance Hub</i> for more details.

Tool	Details
Autonomous AI Database Metrics	The Autonomous AI Database Metrics help you measure useful quantitative data, such as CPU and storage utilization, the number of successful and failed database log in and connection attempts, database operations, SQL queries, and transactions, and so on. You can use metrics data to diagnose and troubleshoot problems with your Autonomous AI Database resources. See Monitor Databases with Autonomous AI Database Metrics for more information such as its prerequisites, usage, and the list of metrics available for Autonomous AI Database on Dedicated Exadata Infrastructure.
Automatic Workload Repository (AWR) and Automatic Database Diagnostic Monitor (ADDM)	Two other commonly used tools are the Automatic Workload Repository (AWR) and the Automatic Database Diagnostic Monitor (ADDM). AWR stores performance related statistics for an Oracle database, and ADDM is a diagnostic tool that analyzes the AWR data on a regular basis, locates root causes of any performance problems, provides recommendations for correcting the problems, and identifies non-problem areas of the system. Because AWR is a repository of historical performance data, ADDM can analyze performance issues after the event, often saving time and resources in reproducing a problem. For instructions on using these tools, as well as detailed information about database performance monitoring and tuning, see Oracle Database 19c Performance Tuning Guide or Oracle Database 26ai Performance Tuning Guide. AWR and ADDM are also available from Performance Hub. See Monitor Database Performance with Performance Hub for more details. For a quick introduction to database performance monitoring and tuning, see Oracle Database 19c 2 Day + Performance Tuning Guide or Oracle Database 26ai 2 Day + Performance Tuning Guide.

Use SQL Tracing on Autonomous AI Database on Dedicated Exadata Infrastructure

You can use SQL Tracing with Autonomous AI Database on Dedicated Exadata Infrastructure to help you identify the source of an excessive database workload, such as a high-load SQL statement in your application.

About SQL Tracing

When an application operation takes longer than expected, getting a trace of all the SQL statements executed as part of this operation with details such as time spent by that SQL statement in the parse, execution, and fetch phases will help you identify and resolve the cause of the performance issue. You can use SQL tracing on an Autonomous AI Database to achieve this.

SQL tracing is disabled by default in Autonomous AI Database. You must enable it to start collecting the SQL tracing data. To trace SQL statements, implement the following tasks as an ADMIN user:

- Start with configuring the database to save SQL Trace files. See [Configure SQL Tracing on Autonomous AI Database](#) for more information.
- Then, enable SQL tracing. See [Enable SQL Tracing on Autonomous AI Database](#).

Note

Enabling SQL tracing may degrade your application performance for the session while the trace collection is enabled. This performance impact is expected due to the overhead of collecting and saving trace data.

- To stop collecting the SQL tracing data, you must disable SQL tracing. See [Disable SQL Tracing](#).
- When you disable SQL tracing, the tracing data collected while the session runs with tracing enabled is written to the SESSION_CLOUD_TRACE view in your session and to a trace file in the bucket, which you configure while setting up SQL tracing. You have two options to view the trace data:
 - View and analyze SQL Trace data in the trace file saved to Cloud Object Store. See [View Trace File Saved to Cloud Object Store on Autonomous AI Database](#) for more information.
 - View and analyze SQL Trace data saved to the view SESSION_CLOUD_TRACE. See [View Trace Data in SESSION_CLOUD_TRACE View on Autonomous AI Database](#) for more information.

Configure SQL Tracing on Autonomous AI Database

To configure your Autonomous AI Database for SQL tracing, do the following:

1. Create a bucket to store trace files in your Cloud Object Storage.

To save the SQL tracing files, the bucket can be in any Cloud Object Store that Autonomous AI Database supports. For example, to create a bucket in Oracle Cloud Infrastructure Object Storage, refer to [Creating a Bucket](#).

Tip

Ensure to pick **Standard** as the **storage tier** when creating your bucket in Oracle Cloud Infrastructure Object Storage because SQL tracing files are only supported with buckets created in the standard storage tier. For information on the Standard Object Storage Tier, see [Understanding Storage Tiers](#).

2. Create a credential for your Cloud Object Storage account using DBMS_CLOUD.CREATE_CREDENTIAL.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
```



```
);END;  
/
```

See `CREATE_CREDENTIAL` Procedure for details on the arguments for the username and password parameters for different object storage services.

3. Set initialization parameters to specify the Cloud Object Storage URL for a bucket for SQL trace files and to specify the credentials to access the Cloud Object Storage.
 - a. Set database property `DEFAULT_LOGGING_BUCKET` to specify the logging bucket on Cloud Object Storage.

For example, if you create the bucket with Oracle Cloud Infrastructure (OCI) Object Storage:

```
SET DEFINE OFF;  
ALTER DATABASE PROPERTY SET  
    DEFAULT_LOGGING_BUCKET = 'https://objectstorage.us-  
ashburn-1.oraclecloud.com/n/namespace-string/b/bucket_name/o/';
```

Where `namespace-string` is the OCI Object Storage namespace and `bucket_name` is the name of the bucket you previously created. See [Understanding Object Storage Namespaces](#) for more information.

See [Regions and Availability Domains](#) for a list of regions.

The Cloud Object Store you use for SQL Tracing files can be any Cloud Object Store that Autonomous AI Database supports.

- b. Set the database property `DEFAULT_CREDENTIAL` to the credential you created in Step 2.

For example:

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

Including the schema name with the credential is required. In this example the schema is `ADMIN`.

Enable SQL Tracing on Autonomous AI Database

Note

Enabling SQL tracing may degrade your application performance for the session while the trace collection is enabled. This performance impact is expected due to the overhead of collecting and saving trace data.

To enable SQL tracing for a database session, , do the following:

1. Optionally, set a client identifier for the application. This step is optional but is recommended. SQL tracing uses the client identifier as a component of the trace file name when the trace file is written to Cloud Object Store.

For example:

```
BEGIN
  DBMS_SESSION.SET_IDENTIFIER('sqlt_test');
END;
/
```

2. Optionally, set a module name for the application. This step is optional but is recommended. SQL tracing uses the module name as a component of the trace file name when the trace file is written to Cloud Object Store.

For example:

```
BEGIN
  DBMS_APPLICATION_INFO.SET_MODULE('modname', null);
END;
/
```

3. Enable the SQL Trace facility.

```
ALTER SESSION SET SQL_TRACE = TRUE;
```

4. Run your workload.

This step involves running the entire application or specific parts of the application. While you run your workload in the database session, SQL tracing data is collected.

5. Disable SQL Tracing.

When you disable SQL tracing the collected data for the session is written to a table in your session and to a trace file in the bucket you configure when you set up SQL tracing.

Disable SQL Tracing

To disable SQL tracing, do the following:

1. Disable the SQL Trace facility.

```
ALTER SESSION SET SQL_TRACE = FALSE;
```

2. Optionally, as needed for your environment, you may want to reset the database property `DEFAULT_LOGGING_BUCKET` to clear the value for the logging bucket on Cloud Object Storage.

For example:

```
ALTER DATABASE PROPERTY SET DEFAULT_LOGGING_BUCKET = '';
```

When you disable SQL tracing, the tracing data collected while the session runs with tracing enabled is copied to a table and sent to a trace file on Cloud Object Store.

View Trace File Saved to Cloud Object Store on Autonomous AI Database

You use SQL trace file data to analyze application performance on Autonomous AI Database. When you disable SQL trace in your database session, data is written to the Cloud Object Store bucket configured with `DEFAULT_LOGGING_BUCKET`.

The SQL Trace facility writes the trace data collected in the session to Cloud Object Store in the following format:

`default_logging_bucket/sqltrace/clientID/moduleName/sqltrace_numID1_numID2.trc`

The components of the file name are:

- `default_logging_bucket`: is the value of the `DEFAULT_LOGGING_BUCKET` database property. See [Configure SQL Tracing on Autonomous AI Database](#) for more information.
- `clientID`: is the client identifier. See [Enable SQL Tracing on Autonomous AI Database](#) for more information.
- `moduleName`: is the module name. See [Enable SQL Tracing on Autonomous AI Database](#) for more information.
- `numID1_numID2`: are two identifiers that the SQL Trace facility provides. The `numID1` and `numID2` numeric values uniquely distinguish each trace file name from other sessions using tracing and creating trace files in the same bucket in the Cloud Object Storage.

When the database service supports parallelism and a session runs a parallel query, the SQL Trace facility can produce multiple trace files with different `numID1` and `numID2` values.

Note

When SQL tracing is enabled and disabled multiple times within the same session, each trace iteration generates a separate trace file in Cloud Object Store. To avoid overwriting previous traces that were generated in the session, subsequently generated files follow the same naming convention and add a numeric suffix to the trace file name. This numeric suffix starts with the number 1 and is incremented by 1 for each tracing iteration thereafter.

For example, the following is a sample generated trace file name when you set the client identifier to "sql_test" and the module name to "modname":

`sqltrace/sqlt_test/modname/sqltrace_5415_56432.trc`

You can run `TKPROF` to translate the trace file into a readable output file.

1. Copy the trace file from Object Store to your local system.
2. Navigate to the directory in which the trace file is saved.
3. Run the `TKPROF` utility from the operating system prompt using the following syntax:

```
tkprof filename1 filename2 [waits=yes|no] [sort=option] [print=n]
[aggregate=yes|no] [insert=filename3] [sys=yes|no] [table=schema.table]
[explain=user/password] [record=filename4] [width=n]
```

The input and output files are the only required arguments.

4. To view online Help, invoke `TKPROF` without arguments.

For information about using the `TKPROF` utility, see *Tools for End-to-End Application Tracing* in Oracle Database 19c SQL Tuning Guide or Oracle Database 26ai SQL Tuning Guide.

View Trace Data in SESSION_CLOUD_TRACE View on Autonomous AI Database

When you enable SQL Tracing, the same trace information that is saved to the trace file on Cloud Object Store is available in the `SESSION_CLOUD_TRACE` view in the session where the tracing was enabled.

While you are still in the database session you can view SQL tracing data in the `SESSION_CLOUD_TRACE` view. The `SESSION_CLOUD_TRACE` view includes two columns: `ROW_NUMBER` and `TRACE`.

```
DESC SESSION_CLOUD_TRACE
```

Name	Null?	Type
ROW_NUMBER		NUMBER
TRACE		VARCHAR2(32767)

The `ROW_NUMBER` specifies the ordering for trace data found in the `TRACE` column. Each line of trace output written to a trace file becomes a row in the table and is available in the `TRACE` column.

After you disable SQL tracing for the session, you can run queries on the `SESSION_CLOUD_TRACE` view.

For example:

```
SELECT trace FROM SESSION_CLOUD_TRACE ORDERBY row_number;
```

The data in `SESSION_CLOUD_TRACE` persists for the duration of the session. After you log out or close the session, the data is no longer available.

If SQL Trace is enabled and disabled multiple times within the same session, `SESSION_CLOUD_TRACE` shows the trace data for all the iterations cumulatively. Thus, re-enabling tracing in a session after previously disabling tracing does not remove the trace data produced by the earlier iteration.

Governance

Tagging Autonomous AI Database Cloud Resources

Tagging is a powerful foundational service for Oracle Cloud Infrastructure (OCI) that enables users to search, control access, and do bulk actions on a set of resources based on the tag.

Importance of Tagging

Using the Oracle Cloud Infrastructure (OCI) tagging system, you can tag resources as per your organizational scheme allowing you to group resources, manage costs, and give insights into usage. Tags also help you build a governance model around security and Maximum Availability Architecture (MAA). As your organization expands its cloud footprint, it can become challenging to keep track of the deployment architectures, security best practices, MAA, application tier, etc. Using metadata tags to identify workload attributes can help keep up with the security and availability of your tenancy without cost overruns.

To enable customers to manage OCI resources securely and cost-effectively, Oracle provides a set of pre-defined tags in line with best practices for tagging resources. These tags are grouped into two namespaces - The *OracleStandard* namespace and the *OracleApplicationName* namespace. You can think of a tag namespace as a container for your tag keys.

Consider a scenario where your organization has multiple cloud resources such as, Autonomous Container Databases, Autonomous AI Database, compute, network and load balancers across multiple compartments in your tenancy. Suppose you wish to track these cloud resources for specific purposes, report on them, or take bulk actions. In that case, you will need a system that lets you group these resources based on different criteria such as environment, criticality, target users, application, etc. You can achieve this by applying appropriate tags to these resources.

For example, you may tag all resources in your development stack with *Oracle-Standard.Environment=Dev* or for a business critical application stack, set *Oracle-Standard.Criticality=High or Extreme*. In the event of service disruptions due to various reasons, you would then be able to quickly identify all OCI resources associated with an application or business function or be able to separate critical and non-critical workloads.

Tagging can also help you deploy optimized configurations based on workload attributes identified via tags. For example, database deployments for the Peoplesoft application require a specific configuration. By setting the *ApplicationName* and *AppMajorVersion* tags, while deploying an Autonomous AI Database, can ensure that the database is configured ready for the particular application (Example: Peoplesoft) out of the box.

Moreover, integration with the Cloud Advisor OCI service can provide you with direct, deep insight into how well your cloud services adhere to the corporate guidelines and help your management govern with a vision. See Cloud Advisor Overview for more details.

Adding Tags

You can tag resources using the Oracle Cloud Infrastructure (OCI) console, command-line interface, or SDK.

There are many cloud resources that can be tagged in an Autonomous AI Database deployment. Exadata Infrastructure, Autonomous Exadata VM Cluster, Autonomous Container Database, and Autonomous AI Database are some of them. Tags can either be applied while creating the resources or modified later. For example, you can apply tags to an Autonomous Container Database (ACD) while provisioning the ACD or add them later from its **Details** page.

See [How Tagging Works](#) for more details on using tags. Tagging integrates with Oracle Cloud Infrastructure authorization system. You can use IAM policy controls to enable delegation or restriction of tag manipulation. See [Authentication and Authorization](#) to learn about the permissions required to work with defined and free-form tags.

✓ Tip

For a "try it out" tutorial that demonstrates implementing tags in Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, refer to *Lab 14: Oracle Standard Tags* in Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop on Oracle LiveLabs.

Your tenancies come with a library of standard tags that would apply to most resources. These tags are currently available as a set of Tag Namespaces that your governance administrators can deploy. OCI best practices recommend applying these tags to all resources a standard tag can be applied to. Besides reporting and governance, OCI service automation can deliver workload-specific optimizations based on standard tag values.

For example, database deployments for the Peoplesoft application require a specific configuration. By setting the appropriate application tag key in the *Oracle-ApplicationName* tag

namespace while deploying an Autonomous AI Database, can ensure that the database is configured ready for the particular application (Example: Peoplesoft) out of the box.

Add tags

[Help](#)

Add tags to organize your resources. [What can I do with tagging?](#)

Tag namespace

OracleAppName

Tag key

PeopleSoft

Tag value

11.2

×

Add tag

Add tags

[Cancel](#)

Oracle Standard Tags

Your tenancy governance administrators can deploy the standard tags at the tenancy level and may also mark certain tags as *required*, thereby enforcing tags on resources in those compartments. The following are the standard tags defined in the namespace called **OracleStandard**.

 **Note**

Oracle Standard tags are applicable for Oracle Public Cloud and Exadata Cloud@Customer deployments only.

Tag Key	Tag Value Options	Description
OracleStandard.Criticality	Extreme High Medium Low	Enables tiering of resource in line with corporate application classification standards. Customer governance can use this tag for reporting and ensuring resources are configured as per the guideline for the tier they belong to. For example, a database resource with <i>OracleStandard.Criticality</i> set to <i>Extreme</i> or <i>High</i> may require the highest availability SLA, and may need to be configured with Autonomous Data Guard.

Tag Key	Tag Value Options	Description
OracleStandard.Environment	Dev Test Prod Pre-prod Staging Trial Sandbox User Testing	Denotes a resource lifecycle. In the case of databases, it helps determine consolidation density, database distribution across containers, set maintenance plans, and manage clones.
OracleStandard.Sensitivity	Public Internal Sensitive Highly Sensitive Extremely Sensitive	An application or database classification tag. <i>OracleStandard.Sensitivity</i> set to <i>Highly Sensitive</i> may indicate that an access control list or certain Network Security Group (NSG) enforcement is mandatory to restrict access.
OracleStandard.Regulation	Refer to List of Compliance Regulations for values.	Denotes one or more compliance regulations that a resource must adhere to. Tag administrators may add values to the list from the OCI <i>Governance and Administration</i> console. Refer to Using Predefined Values for more details.
OracleStandard.TargetUsers	Public Customers Partners Company Division Department Workgroup	Denotes the end users of a resource. Another form of resource classification that helps determine target users and allow governance teams to set corporate standards based on user or application type.
OracleStandard.EndUserCount	1 10 100 1000 10000 100000 1000000 1000000 10000000	An approximate count of end-users. This tag helps determine the number of users impacted or blast radius during an availability or security event. This also help prioritize recovery efforts in the event of major outages affecting a large number of cloud resources.
OracleStandard.OwnerEmail	Free form tag. For example: john.smith@acme.com or app_support_grp@acme.com	Denotes the email address of the resource owner.

Tag Key	Tag Value Options	Description
OracleStandard.Org	HR, Finance, Marketing, Sales, Legal, R&D, Customer Support, Internal Support, Manufacturing	Identifies the customer's line of business or department that owns or uses the resource. This may help with cost aggregation reports and determining usage across business units. Tag administrators may add relevant values to the list from the OCI <i>Governance and Administration</i> console. Refer to Using Predefined Values for more details.
OracleStandard.CostCenter	12345, WebMarketing	Freeform field for cost center.
OracleStandard.RecoveryTimeObjectiveMinutes	0-10080	Time in minutes. Denotes the maximum time within which the resource is required to recover from a failure.
OracleStandard.RecoveryPointObjectiveMinutes	0-1440	Time in minutes. Maximum data loss tolerance for a data store resource such as a database or a storage device.

List of Compliance Regulations

The table below lists the valid values that you can apply for the *OracleStandard.Regulation* tag.

Regulation	Description
PCI DSS	Payment Card Industry Data Security Standard
HIPAA	Health Insurance Portability and Accountability Act
ISO	International Standards Organization
SOC1	System and Organization Controls 1
SOC 2	System and Organization Controls 2
FedRamp	Federal Risk and Authorization Management Program
GLBA	Gramm–Leach–Bliley Act
CCPA	California Consumer Privacy Act
SOX	Sarbanes Oxley
NIST	National Institute of Standards and Technology - Cyber Security
FISMA	Federal Information Security Management
HITECH	Health Information Technology for Economic and Clinical Health Act
FERPA	Family Educational Rights and Privacy Act (Student privacy)
FACTA	Fair and Accurate Credit Transaction Act
Texas HB300	Texas Medical Records Privacy Act
CIS	Center for Internet Security
CJIS	Criminal Justice Information Services Security Policy
C-TPAT	Customs-Trade Partnership Against Terrorism

Regulation	Description
COPPA	Children's Online Privacy Protection Act
PIPED Act, or PIPEDA	Personal Information Protection and Electronic Documents Act
GDPR	General Data Protection Regulation
PIPL	Personal Information Protection Law

Oracle Application Name Tags

The following are the application name tags defined in the namespace called **Oracle-ApplicationName**.

Tag Key	Tag Value Options	Description
Hyperion	11.2	Denotes the version of the Hyperion application.
	11.1	
JD_Edwards	9.2	Denotes the version of the JD Edwards application.
	9.1	
	9.0	
Oracle_E-Business_Suite	12.2	Denotes the version of the Oracle E-Business Suite application.
	12.1	
	12.0	
	11i	
PeopleSoft	9.2	Denotes the version of the Peoplesoft application.
	9.1	
Siebel	8.2	Denotes the version of the Siebel application.
	8.1	
Other_Oracle_Application	Free form tag in string format.	Can be used to denote any application other than those listed above. You can enter the application name as a string value.

Manage Resource Availability with Compartment Quotas

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure supports the compartment quotas feature of Oracle Cloud Infrastructure.

Using this feature you can restrict, on a compartment by compartment basis, the availability of the following dedicated infrastructure resources in your tenancy:

- ECPUUs available to dedicated deployments of Autonomous AI Lakehouse databases.
- OCPUUs available to dedicated deployments of Autonomous AI Lakehouse databases.
- Total data storage available to dedicated deployments of Autonomous AI Lakehouse databases.
- ECPUUs available to dedicated deployments of Autonomous AI Transaction Processing databases.
- OCPUUs available to dedicated deployments of Autonomous AI Transaction Processing databases.

- Total data storage available to dedicated deployments of Autonomous AI Transaction Processing databases.

Note

Quotas are enforced only for provisioning databases and manually scaling their allocated CPUs up or down.

You can manage the compartment quotas to limit the availability of a given resource in a given compartment using the **set**, **unset** and **zero** policy statements. These policy statements are written in a simple declarative language that is similar to the IAM policy language.

- **set**: Sets the maximum number of cloud resources that can be used for a compartment. For example, the following statement sets `atp-dedicated-ecpu-count` quota to 40 ECPUs in the compartment `MyCompartment` :

```
set database quota atp-dedicated-ecpu-count to 40 in compartment
MyCompartment
```

- **unset**: Resets quotas back to the default service limits. For example, the following statement resets the quota for `atp-dedicated-ecpu-count` in the entire tenancy:

```
unset database quota atp-dedicated-ecpu-count in tenancy
```

- **zero**: Removes access to a cloud resource for a compartment. For example, the following statement disallows use of outbound email or notifications in the compartment `MyCompartment`:

```
Zero email-delivery quotas in compartment MyCompartment
Zero notifications quotas in compartment MyCompartment
```

See [Sample Quotas](#) for more such examples and [Quota Policy Syntax](#) for an explanation of the anatomy of quota statements.

Compartment quotas are available on the OCI Console's **Limits, Quotas and Usage** page, which you can use to automatically generate quota statements. As a prerequisite, ensure you have the appropriate permissions to create quotas. See [Policy Reference for the Quotas Service](#) for more details.

Quotas for Autonomous AI Database on Oracle Public Cloud

Family name: database

Name	Scope	Description
<code>adw-dedicated-ecpu-count</code>	Availability domain	Number of Autonomous AI Lakehouse ECPUs for databases using dedicated Exadata infrastructure. (See note about "n/a" values on the Limits, Quotas and Usage page of the Console.)

Name	Scope	Description
adw-dedicated-ocpu-count	Availability domain	Number of Autonomous AI Lakehouse OCPUs for databases using dedicated Exadata infrastructure. (See note about "n/a" values on the Limits, Quotas and Usage page of the Console.)
adw-dedicated-total-storage-tb	Availability domain	Amount of storage (in TB) for Autonomous AI Lakehouse databases using dedicated Exadata infrastructure. (See note following this table about "n/a" values on the Limits, Quotas and Usage page of the Console.)
atp-dedicated-ecpu-count	Availability domain	Number of Autonomous AI Transaction Processing ECPUs for databases using dedicated Exadata infrastructure. (See note following this table about "n/a" values on the Limits, Quotas and Usage page of the Console.)
atp-dedicated-ocpu-count	Availability domain	Number of Autonomous AI Transaction Processing OCPUs for databases using dedicated Exadata infrastructure. (See note following this table about "n/a" values on the Limits, Quotas and Usage page of the Console.)
atp-dedicated-total-storage-tb	Availability domain	Amount of storage (in TB) for Autonomous AI Transaction Processing databases using dedicated Exadata infrastructure. (See note following this table about "n/a" values on the Limits, Quotas and Usage page of the Console.)

Quotas for Autonomous AI Database on Exadata Cloud@Customer

Family name: database

Name	Scope	Description
adw-exacc-ecpu-count	Availability domain	Number of Autonomous AI Lakehouse ECPUs for databases using Exadata Cloud@Customer. (See note about "n/a" values on the Limits, Quotas and Usage page of the Console.)
adw-exacc-ocpu-count	Availability domain	Number of Autonomous AI Lakehouse OCPUs for databases using Exadata Cloud@Customer. (See note about "n/a" values on the Limits, Quotas and Usage page of the Console.)

Name	Scope	Description
adw-exacc-total-storage-tb	Availability domain	Amount of storage (in TB) for Autonomous AI Lakehouse databases using Exadata Cloud@Customer. (See note following this table about "n/a" values on the Limits, Quotas and Usage page of the Console.)
atp-exacc-ecpu-count	Availability domain	Number of Autonomous AI Transaction Processing ECPUs for databases using Exadata Cloud@Customer. (See note following this table about "n/a" values on the Limits, Quotas and Usage page of the Console.)
atp-exacc-ocpu-count	Availability domain	Number of Autonomous AI Transaction Processing OCPUs for databases using Exadata Cloud@Customer. (See note following this table about "n/a" values on the Limits, Quotas and Usage page of the Console.)
atp-exacc-total-storage-tb	Availability domain	Amount of storage (in TB) for Autonomous AI Transaction Processing databases using Exadata Cloud@Customer. (See note following this table about "n/a" values on the Limits, Quotas and Usage page of the Console.)

Continuous Availability

Code for Continuous Availability

Your applications achieve *continuous availability* when planned maintenance, unplanned outages, and load imbalances of the database are hidden from the application. A combination of application best practice, simple configuration, and Oracle Autonomous AI Database ensure that your applications are continuously available.

The best approach for hiding planned maintenance activities from your applications is to transparently drain work from each database workload location prior to the maintenance window for that workload location. Oracle's connection pools and mid-tiers, including the WebLogic Server, Oracle Universal Connection Pool (UCP), OCI Session pool and ODP.NET Unmanaged Provider are Fast Application Notification (FAN) aware and therefore are notified before database services are scheduled to move to allow graceful draining of work before maintenance. FAN notification automatically triggers closing idle connections, opening new connections in the new service location, and allows a configurable time for active work to complete in the soon-to-be-shutdown service location. The major third-party JDBC mid-tiers, such as IBM WebSphere, allow for the same behavior when configured with UCP. For JDBC-based applications that cannot use UCP, Oracle provides solutions using Oracle Drivers and connection tests.

In order to hide unplanned outages resulting from a component or communication failure Oracle provides:

- **Notification.** FAN is the first step to hiding outages. FAN notifies clients and breaks them out of their current network wait when an outage occurs. This avoids stalling applications for long network waits. Importantly, FAN also invokes rebalancing of sessions when services are available again.
- **Recovery.** After the client is notified, Transparent Application Continuity (TAC) or Application Continuity (AC) re-establish a connection to a new workload location (another database instance in the Real Application Clusters (RAC) configuration running the database) and replays in-flight (uncommitted) work when possible. By replaying in-flight work on the new location, the application can usually continue executing without knowing that any failure happened.

TAC or AC also executes during planned maintenance for those sessions that do not drain (complete their current database operation) during the allocated drain interval.

Application Configuration Checklist

You make your application continuously available by following these guidelines:

- [Connect Using Database Services](#)
- [Configure Connection String for High Availability](#)
- [Use Fast Application Notification \(FAN\)](#)
- [Use Recommended Practices to Allow Draining](#)
- [Enable Transparent Application Continuity \(TAC\) or Application Continuity \(AC\)](#)

✓ Tip

See Continuous Availability for Applications on ATP-Direct White Paper to know about the best practices in implementing continuous availability for applications using an Autonomous AI Database.

Connect Using Database Services

Database services provide transparency for the underlying infrastructure: FAN, connection data, Transparent Application Continuity (TAC), Application Continuity (AC), switchover, consumer groups and many other features and operations are predicated on the use of services.

Autonomous AI Database on Dedicated Exadata Infrastructure offers several pairs of predefined database services to choose from, as described in Predefined Database Service Names for Autonomous AI Databases. All provide FAN and draining, and the two transaction processing pairs have TAC enabled by default. An API is available to change the TAC or AC settings on all predefined services (see Enable Service Attributes for Failover).

Configure Connection String for High Availability

Oracle recommends the Connection String configuration shown below when connecting to Oracle Autonomous AI Database. Connect strings embedded in the Oracle-supplied `tnsnames.ora` file are configured in this fashion. Do not use Easy Connect Naming on the client because such connections have no high-availability capabilities.

Use this TNS for all Oracle clients version 12.2 or higher:

```
alias =
(DESCRIPTION =
(CONNECT_TIMEOUT= 120)(RETRY_COUNT=20)(RETRY_DELAY=3)(TRANSPORT_CONNECT_TIMEOUT=3)
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST=scan-host)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = service-name)))
```

Use the following for JDBC connections using Oracle driver version 12.1 or earlier

```
alias =
(DESCRIPTION =
(CONNECT_TIMEOUT= 15)(RETRY_COUNT=20)(RETRY_DELAY=3)
(ADDRESS_LIST =
  (LOAD_BALANCE=on)
  (ADDRESS = (PROTOCOL = TCP)(HOST=scan-host)(PORT=1521)))
(CONNECT_DATA=(SERVICE_NAME = service-name)))
```

Use Fast Application Notification (FAN)

FAN provides immediate notification to an application in the event of an outage or resumption of service. Without FAN, applications can hang on TCP/IP timeout following hardware and network failures, and omit to rebalance when resources resume. All Oracle pools and all Oracle application servers use FAN. Third-party JAVA application servers can use UCP to enable FAN.

No application changes are required to use FAN. These are configuration changes only.

For continuous service during planned maintenance, use FAN with:

- Oracle pools or
- UCP with third-party JDBC application servers or
- The latest Oracle client drivers

For continuous service during unplanned outages, use FAN with:

- Transparent Application Continuity or
- Application Continuity

FAN Coverage

FAN events are integrated with:

- Oracle Fusion Middleware and Oracle WebLogic Server
- Oracle Data Guard Broker
- Oracle JDBC Universal Connection Pool or Driver for both JDBC thin and Oracle Call Interface (OCI) interfaces
- ODP.NET Connection Pool for Unmanaged and Managed Providers
- Oracle Tuxedo
- SQL*Plus
- Oracle Database drivers for languages such as Python, Node.js and PHP
- Global Data Services
- Third party JDBC application servers using Oracle JDBC Universal Connection Pool
- Listeners

To Enable FAN in the Client

Use the TNS alias shown in [Configure Connection String for High Availability](#). This connection string is used to auto-configure the Oracle Notification Service (ONS) subscription at the client for FAN-event receipt when using an Oracle Database 12c or later client driver. ONS provides a secure communication path between the database tier and the client-tier allowing the client to be notified of service availability (components stopping or starting) as well as runtime load balancing advice for better work placement during normal operation.

Depending on the client, enable FAN in the application configuration properties as follows:

- Universal Connection Pool or JDBC thin driver (starting 12.2)
Set the property `FastConnectionFailoverEnabled`.
- WebLogic Active GridLink for Oracle
RAC FAN and Fast Connection Failover are enabled by default.
- Oracle WebLogic Server, IBM WebSphere, IBM Liberty, Apache Tomcat, Red Hat WildFly (JBoss), JDBC Applications
Use Universal Connection Pool as a connection pool replacement.
- ODP.Net clients (Managed and Unmanaged Providers)
Set `"HA events = true;pooling=true"` in the connect string if using ODP.Net 12.1 or earlier.
- Oracle Call Interface (OCI) clients and OCI-based drivers
Oracle Call Interface (OCI) clients without native settings can use an `oraacces.xml` file and set events to true.

Python, Node.js and PHP have native options. In Python and Node.js you can set an events mode when creating a connection pool. In PHP, edit `php.ini` add the entry `oci8.events=on`.
- SQL*Plus
FAN is enabled by default.

The predefined database services offer TCPS connections that use TLS wallet-based authentication. Depending upon the type of application (JDBC or Oracle Call Interface), the wallet configuration must follow particular rules, as described in [Configure Clients for FAN Including Optional Wallets](#).

Use Recommended Practices to Allow Draining

Best practice for application usage is to check out connections for the time that they are needed, and then check them back in to the pool when the current action is complete. This is important to achieve good performance, for the rebalancing of work at runtime, and during maintenance windows for draining the work.

Oracle recommends using a FAN-aware Oracle connection pool for hiding planned maintenance. There is no impact to users when your application uses an Oracle Pool with FAN and returns connections to the pool between requests. You do not need to make any application changes to use FAN. When an Oracle connection pool receives the FAN event for planned downtime, it marks all connections at the instance to be drained. Immediately, checked-in connections are closed so that they are not re-used. As in-use connections are returned to the pool they are closed. This allows all connections to be closed gracefully over time.

If you are using a third-party, Java-based application server, then the most effective method to achieve draining and failover is to replace the pooled data source with UCP. Many application servers support this approach, including Oracle WebLogic Server, IBM WebSphere, IBM Liberty, Apache Tomcat, Red Hat WildFly (JBoss), Spring, Hibernate, and others. White papers from Oracle and other providers, such as IBM, describe how to use UCP with these application servers. Using UCP as the data source allows UCP features such as Fast Connection Failover, Runtime Load Balancing, Application Continuity and Transparent Application Continuity to be used with full certification.

Enable Transparent Application Continuity (TAC) or Application Continuity (AC)

TAC transparently tracks and records session and transactional state so that a database session can be recovered following recoverable outages. The two transaction processing pairs of predefined database services have TAC enabled by default.

AC is customizable, allowing you to choose to replay side effects or to add complex callbacks at failover that TAC does not allow. Use AC if you are using Oracle 12c drivers (JDBC-thin or Oracle Call Interface), or you want to customize with side effects or callbacks, or have an application that uses state such as session duration temp tables and does not clean up across requests.

Steps for Using Transparent Application Continuity

- If you need to enable Transparent Application Continuity in the database service you are using, see [Enable Service Attributes for Failover](#).
- Use one of the following supported clients.

Oracle strongly recommends that you use the latest client drivers. Oracle Database 19c client drivers and later provide full support for TAC.

- Oracle JDBC Replay Driver 18c or later. This is a JDBC driver feature provided with Oracle Database 18c for Application Continuity.
 - Oracle Universal Connection Pool (UCP) 18c or later with Oracle JDBC Replay Driver 18c or later.
 - Oracle WebLogic Server Active GridLink, or third-party JDBC application servers using UCP with Oracle JDBC Replay Driver 18c or later.
 - Java connection pools or standalone Java applications using Oracle JDBC Replay Driver 18c or later.
 - Oracle Call Interface Session Pool 19c or later.
 - SQL*Plus 19c (19.3) or later
 - ODP.NET pooled, Unmanaged Driver 18c or later ("Pooling=true" default in 12.2 and later).
 - Oracle Call Interface based applications using 19c OCI driver or later.
- Return Connections to the Connections Pool.

You do not need to make any changes to your application for identifying request boundaries if the application uses connections:

- from the Oracle connection pools or
- from the Oracle JDBC Replay Driver 18c or later or
- from the Oracle Call Interface based applications using 19c or later

When using connection pools, the application should return the connection to the pool on completion of each request. Oracle recommends that an application checks out a

connection only for the time it needs it. Holding a connection when not in use is not good practice. Transparent Application Continuity with the listed drivers also detects where boundaries can be added and makes its own boundaries.

- Use `FAILOVER_RESTORE`

Enabling Transparent Application Continuity automatically restores preset session states. If you find that you need preset session states in addition to the standard set, then you can register a callback or UCP label to restore these states. If you find that you need complex session states, such as temporary tables or `SYS_CONTEXT`, restored, then use Application Continuity that offers this flexibility.

- Enable Mutable Use in the Application

Mutable functions are functions that can return a new value each time they are executed. Support for keeping the original results is provided for `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and `sequence.NEXTVAL`. Application Continuity 19c and later automatically `KEEPS` mutables for SQL. If your application uses, or is sensitive to, mutable functions, then a DBA must issue the `GRANT KEEP` privilege. When `KEEP` privilege is granted, replay applies the original function result at replay. For example:

```
SQL> GRANT [KEEP DATE TIME | KEEP SYSGUID] ... TO USER
```

```
SQL> GRANT KEEP SEQUENCE mySequence TO myUser ON sequence.object
```

- Side Effects Are Disabled

A side effect is an external action such as sending mail, transferring files or using TCP. Transparent Application Continuity detects side effects and does not replay them. If you want the side effects replayed, then use Application Continuity that allows this extra flexibility.

Steps for Using Application Continuity

- If you need to enable Application Continuity in the database service you are using, see [Enable Service Attributes for Failover](#).
- Use one of the following supported clients.
 - Oracle JDBC Replay Driver 12c or later. This is a JDBC driver feature provided with Oracle Database 12c for Application Continuity.
 - Oracle Universal Connection Pool (UCP) 12c or later with Oracle JDBC Replay Driver 12c or later.
 - Oracle WebLogic Server Active GridLink and third-party JDBC application servers using UCP with Oracle JDBC Replay Driver 12c or later.
 - Java connection pools or standalone Java applications using Oracle JDBC Replay Driver 12c or later with Request Boundaries or Pooled Data Source.
 - Applications and language drivers using Oracle Call Interface Session Pool 12c Release 2 or later.
 - `SQL*Plus` 19.3 or later.
 - `ODP.NET` pooled, Unmanaged Driver 12c Release 2 or later (`"Pooling=true"; "Application Continuity=true"` default in 12.2 and later)
- Return Connections to the Connections Pool.

You do not need to make any changes to your application for identifying request boundaries if the application is using an Oracle connection pool or a third-party JDBC pool that supports request boundaries. It is best practice to use an Oracle pool and return the connections to that pool between requests. Oracle recommends that an application checks

out a connection only for the time it needs it. Holding a connection when not in use is not good practice, and it will prevent transparent planned maintenance.

- Use `FAILOVER_RESTORE`

Most common states are restored automatically with `FAILOVER_RESTORE=LEVEL1`. If your application presets session states in addition to the standard set, then you must register a callback, or UCP label, to restore these states. Set `FAILOVER_RESTORE=LEVEL1` on the service and use:

- Connection Initialization Callback for Java or the (older) TAF Callback for Oracle Call Interface or
- Universal Connection Pool or WebLogic Server Connection Labelling

- Enable Mutable Use in the Application

Mutable functions are functions that can return a new value each time they are executed. Support for keeping the original results is provided for `SYSDATE`, `SYSTIMESTAMP`, `SYS_GUID`, and `sequence.NEXTVAL`. Application Continuity 19c and later automatically `KEEPS` mutables for SQL, so no action is required. If you need mutables for PL/SQL, then a DBA must issue the `GRANT KEEP` privilege. When `KEEP` privilege is granted, replay applies the original function result at replay. For example:

```
SQL> GRANT [KEEP DATE TIME | KEEP SYSGUID] ... TO USER
```

```
SQL> GRANT KEEP SEQUENCE mySequence TO myUser ON sequence.object
```

- Decide If You Want to Replay Side Effects

A side effect is an external action such as sending mail, transferring files or using TCP. With Application Continuity, side effects are replayed unless the application specifies otherwise. If a request has an external action that should not be replayed, then that request can use a connection that does not have Application Continuity enabled, or replay can be disabled for that request using the `disableReplay()` API for Java or `OCIRequestDisableReplay()` for Oracle Call Interface. If you do not wish to replay all side effects, use Transparent Application Continuity.

Configure Drivers for Continuous Availability

This article describes how to configure drivers for continuous availability.

Configure JDBC Thin Driver

1. Ensure all recommended patches are applied at the client. Refer to the MOS note *Client Validation Matrix for Application Continuity* (Doc ID 2511448.1).
2. Use JDBC Statement Cache for Coverage and Performance

For best coverage and performance, use the JDBC driver statement cache in place of an application server statement cache. This allows the driver to know that statements are closed and memory to be freed at the end of requests.

To use the JDBC statement cache, use connection property `oracle.jdbc.implicitStatementCacheSize` (`OracleConnection.CONNECTION_PROPERTY_IMPLICIT_STATEMENT_CACHE_SIZE`). The statement cache is per connection. The value for the cache size matches your number of open cursors. For example:

`oracle.jdbc.implicitStatementCacheSize=nnn` where *nnn* is typically between 10 and 100 and is equal to the number of open cursors your application maintains.

3. Tune the Garbage Collector

For many applications the default Garbage Collector tuning is sufficient. For applications that return and keep large amounts of data you can use higher values, such as 2GB or larger. For example:

```
java -Xms3072m -Xmx3072m
```

It is recommended to set the memory allocation for the initial Java heap size (ms) and maximum heap size (mx) to the same value. This prevents using system resources on growing and shrinking the memory heap.

4. JDBC Concrete Classes

For JDBC applications, Oracle does not support deprecated `oracle.sql` concrete classes BLOB, CLOB, BFILE, OPAQUE, ARRAY, STRUCT or ORADATA. (See MOS note 1364193.1, *New JDBC Interfaces*). Use `ORAchk -acchk` on the client to know if an application passes. The list of restricted concrete classes for JDBC Replay Driver is reduced to the following starting with Oracle JDBC-thin driver version 18c and later: `oracle.sql.OPAQUE`, `oracle.sql.STRUCT`, `oracle.sql.ANYDATA`

5. Configure Fast Connection Failover (FCF)

This corresponds to enabling subscription to FAN events.

For client drivers 12c and later:

- Use the recommended URL for auto-ons
- Check that `ons.jar` (plus optional `WALLET` jars, `osdt_cert.jar`, `osdt_core.jar`, `oraclepki.jar`) are on the CLASSPATH
- Set the pool or driver property `fastConnectionFailoverEnabled=true`
- UCP (recommended) or third-party connection pools
- Open port 6200 for ONS (6200 is the default port, a different port may have been chosen)

For client drivers prior to 12c use the addresses provided:

- Set `oracle.ons.nodes =XXX01:6200, XXX02:6200, XXX03:6200`

Configure OCI (Oracle Call Interface) Driver

1. Ensure all recommended patches are applied at the client. Refer to the MOS Note *Client Validation Matrix for Application Continuity* (Doc ID 251148.1).
2. Replace `OCIStmtPrepare` with `OCIStmtPrepare2`. `OCIStmtPrepare()` has been deprecated since 12.2. All applications should use `OCIStmtPrepare2()`. TAC and AC allow `OCIStmtPrepare` but do not replay this statement.
3. To use FAN for OCI-based applications, do the following:
 - `aq_ha_notifications` is preset on the services
 - Use the recommended Connection String for auto-ons
 - Set `auto_config`, `events`, and `wallet_location` (optional) in `oraaccess.xml`, as described in [Configure Clients for FAN Including Optional Wallets](#).
 - Link the application with the O/S client thread library
 - Open port 6200 for ONS (6200 is the default port, a different port may have been chosen). For client drivers prior to 12c use the addresses provided in `oraaccess.xml`.

Configure ODP.NET Unmanaged Provider Driver

1. Ensure all recommended patches are applied at the client. Refer to the MOS Note *Client Validation Matrix for Application Continuity* (Doc ID 251148.1).
2. To use FAN for Oracle Call Interface based applications, do the following:
 - aq_ha_notifications is preset on the services
 - Use the recommended Connection String for auto-ons
 - Set onsConfig and wallet_location (optional) in oraaccess.xml, as described in [Configure Clients for FAN Including Optional Wallets](#).
 - Open port 6200 for ONS (6200 is the default port, a different port may have been chosen)
 - Set FAN, in the connection string: "user_id=oracle; password=oracle; data source=HA; pooling=true; HA events=true;"
 - (optional) Set Runtime Load Balancing, also in the connection string: "user_id=oracle; password=oracle; data source=HA; pooling=true; HA events=true; load balancing=true;"

Enable Service Attributes for Failover

Transparent Application Continuity is enabled by default for the predefined services tp_tls, tp, tpurgent_tls and tpurgent, so you need to do nothing if you are using one of them. Additionally for them, the DEFAULT value for FAILOVER_RESTORE is AUTO.

You can change the failover type offered on your service by using the package DBMS_APP_CONT_ADMIN. Use this API to enable Application Continuity, Transparent Application Continuity or Transparent Application Failover (TAF), or to disable failover altogether. New sessions will use the new failover type. You must be an administrator to use these procedures.

To enable Transparent Application Continuity for a service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAC('databaseid_high.atp.oraclecloud.com');
```

To enable Application Continuity for a service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_AC('databaseid_tpurgent.atp.oraclecloud.com');
```

To enable TAF SELECT for a service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAF('databaseid_low.atp.oraclecloud.com');
```

To enable TAF BASIC for a service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAF('databaseid_medium.atp.oraclecloud.com',  
'SESSION');
```

To disable failover for a service:

```
execute DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER('databaseid_high.atp.oraclecloud.com');
```

If you wish to use TAF without modifying services, use the older client-side configuration for TAF in your connection string:

```
(FAILOVER_MODE=(TYPE=select)(METHOD=basic)(OVERRIDE=TRUE))
```

Configure Service Attributes for Maintenance

When planned maintenance starts, sessions that need to be drained are marked to be drained. Idle sessions are released gradually. Active sessions are drained when the work executing in that session completes. Draining of sessions is in wide use with Oracle connection pools and mid-tiers configured for Fast Application Notification (FAN). Starting with Oracle Database 18c, the database itself drains sessions when databases are stopped or relocated. Draining is always the best solution for hiding planned maintenance.

Failover solutions such as Application Continuity are the fallback when work will not drain in the time allocated.

A drain timeout, in seconds, and a stop option can be set on a service using the package `DBMS_APP_CONT_ADMIN`. You must be an administrator to use these procedures.

To adjust drain timeout to 600 seconds and set stop option to IMMEDIATE:

```
execute DBMS_APP_CONT_ADMIN.SET_DRAINING('databaseid_high.atp.oraclecloud.com', 600,
'IMMEDIATE' );
```

To adjust only drain timeout :

```
execute DBMS_APP_CONT_ADMIN.SET_DRAINING('databaseid_tpurgent.atp.oraclecloud.com', 600);
```

Transparent Application Continuity (TAC) is a failover option set using the procedure `DBMS_APP_CONT_ADMIN.ENABLE_TAC`. The `ENABLE_TAC` procedure takes three parameters: `SERVICE_NAME` is the service name to change, `FAILOVER_RESTORE`, set to `AUTO` to select Transparent Application Continuity (TAC), and `REPLAY_INITIATION_TIMEOUT` is the replay timeout that specifies how many seconds after a request is submitted to allow that request to replay.

For example, as the ADMIN user, to enable Transparent Application Continuity (TAC) for the TP service with the replay timeout set to 20 minutes:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAC('databaseid_tp.atp.oraclecloud.com', 'AUTO',
1200);
```

Application Continuity (AC) is a failover option set using the procedure `DBMS_APP_CONT_ADMIN.ENABLE_AC`. The `ENABLE_AC` procedure takes three parameters: `SERVICE_NAME` is the service name to change, `FAILOVER_RESTORE`, set to `LEVEL1` to select Application Continuity(AC), and `REPLAY_INITIATION_TIMEOUT`, is the replay timeout that specifies how many seconds after a request is submitted to allow that request to replay.

For example, as the ADMIN user, to enable Application Continuity for the TPURGENT service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_AC('databaseid_tpurgent.adb.oraclecloud.com',
'LEVEL1', 600);
```

Configure Clients for FAN Including Optional Wallets

Wallet-based authentication is an option for FAN when using Autonomous AI Database. Use the same wallet as for the TNS connection.

For JDBC Applications

1. Ensure the following jar files are present in the application's CLASSPATH:
(`ons.jar`, `osdt_cert.jar`, `osdt_core.jar`, `oraclepki.jar`)
2. Specify the wallet for FAN in one of the following ways:

- To use auto-configured ONS with wallets, set the following Java system properties:

```
"-Doracle.ons.walletfile=/replace this with host path/onswallet"
```

```
"-Doracle.ons.walletpassword=myONSWalletPassword"
```

Note that these cannot be set on a per-pool or per-connection basis.

- To explicitly set ONS do one of the following:
 - Set explicitly using an UCP XML Configuration file. For example:

```
<!--?xml version="1.0" encoding="UTF-8"? -->
<ucp-properties>
  <connection-pool
    connection-pool-name="UCP_pool1"
    user="dbuser"
    password="dbuserpasswd"
    connection-factory-class-name="oracle.jdbc.pool.OracleDataSource"
    initial-pool-size="10"
    min-pool-size="5"
    max-pool-size="15"
    validate-connection-on-borrow="true"
    connection-wait-timeout="900"
    max-connections-per-service="50"
    sql-for-validate-connection="select 1 from dual"
    url="jdbc:oracle:thin:@(DESCRIPTION =(CONNECT_TIMEOUT= 120)
      (RETRY_COUNT=20) (RETRY_DELAY=3) (TRANSPORT_CONNECT_TIMEOUT=3)(ADDRESS_LIST
      =(LOAD_BALANCE=on)(ADDRESS = (PROTOCOL = TCP)(HOST=primary-scan)(PORT=1521)))
      (ADDRESS_LIST =(LOAD_BALANCE=on)(ADDRESS = (PROTOCOL = TCP)(HOST=standby-
      scan)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME = MY-SERVICE)))"
    fastConnectionFailoverEnabled="true"
    onsConfiguration="nodes=primary-scanhost:6200,secondary-
    scanhost:6200\nwalletfile=/replace_with_host_path/
    onswallet\nwalletpassword=myWalletPassword">
  </connection-pool>
</ucp-properties>
```

- Set programmatically from within UCP, using a call to `setONSConfiguration()`, for example:

```
pds.setONSConfiguration("nodes=primary-scanhost:6200,secondary-
scanhost:6200\nwalletfile=/replace_this_with_host_path/
onswallet\nwalletpassword=myWalletPassword");
```

For Oracle Call Interface (OCI) Applications Using Oracle Driver Version 12.2 or More Recent

Add the following to the `<default_parameters>` section of the `oraaccess.xml` file:

```
<default_parameters>
  (Other settings may be present in this section)
  <events>
    true
  </events>
  <ons>
    <auto_config>true</auto_config>
    <wallet_location>/my_path/onswallet</wallet_location>
  </ons>
</default_parameters>
```

The `<wallet_location>` path should be the name of the directory containing the wallet.

Other parameters may be set in the `ons` section of `oraaccess.xml`, including `<hosts>`, `<max_connections>`, and `<subscription_wait_timeout>`.

Drivers that support native event setting controls may omit the `<events>` section and use the driver setting instead.

By default connections will be established to the database even if ONS fails. If you prefer connections to fail in this scenario, you can add a section to the same level as `<events>` and `<ons>`:

```
<fan>
  <subscription_failure_action>
    error
  </subscription_failure_action>
</fan>
```

Place the `oraaccess.xml` file in the same directory as the `tnsnames.ora` and `sqlnet.ora` network files. For example, when using Oracle Instant Client these files might be in the default directory `network/admin`. Alternatively, all network configuration files can be put in another accessible directory. Then set the environment variable `TNS_ADMIN` to that directory name.

ODP.Net Managed Provider

Use the `application.config` file to specify ONS configuration and wallet location. For example:

```
<oracle.manageddataaccess.client>
  <version number="*">
    <onsConfig mode="remote">
      <settings>
        <setting name="Protocol" value="TCPS" />
        <setting name="WALLET_LOCATION" value="C:\myPath\ONS_SSLWallet" />
      </settings>
      <ons database="atp01db">
        <add name="nodeList" value="racNode1:6205, racNode2:6205, racNode3:6205" />
      </ons>
    </onsConfig>
  </version>
</oracle.manageddataaccess.client>
```

Compute Management

Compute Models in Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database on Dedicated Exadata Infrastructure offers two **compute models** while configuring your Autonomous AI Database resources. They are:

- **ECPU:** An ECPU is an abstracted measure of compute resources. ECPUs are based on the number of cores elastically allocated from a pool of compute and storage servers. You need at least 2 ECPUs to provision an Autonomous AI Database.

While provisioning a new database, cloning an existing database, and scaling up or down the CPU resources of an existing database, the CPU count defaults to 2 ECPUs, in increments of 1. For example, the next available number of ECPUs above 2 is 3.

You can create Autonomous AI Database for Developers instances on ECPU based container databases. They are free Autonomous AI Databases that developers can use to

build and test new applications. See Autonomous AI Database for Developers for more details.

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

Note

OCPU is a legacy billing metric and has been retired for Autonomous AI Database on Dedicated Exadata Infrastructure. Oracle recommends using ECPUs for all new and existing Autonomous AI Database deployments. See [Oracle Support Document 2998755.1](#) for more information.

While provisioning a new database, cloning an existing database, and scaling up or down the CPU resources of an existing database:

- The CPU count defaults to 1 OCPU, in increments of 1. For example, the next available number of OCPUs above 1 is 2.
- For databases that do not need an entire OCPU, you can assign OCPUs from 0.1 to 0.9 in increments of 0.1 OCPUs. This allows you to overprovision CPU and run more databases on each infrastructure instance. Refer to CPU Overprovisioning for more details.

The Autonomous Exadata VM Cluster's compute type applies to all its Autonomous Container Databases and Autonomous AI Database instances.

Compute Management

Autonomous AI Database instances are deployed into an Autonomous Exadata VM Cluster (AVMC) and into one of its child Autonomous Container Databases (ACD). Exadata Infrastructures are capable of running multiple AVMCs. The CPUs that you allocate while provisioning an AVMC resource will be the **total CPUs** available for its Autonomous AI Databases. When you create multiple AVMCs, each AVMC can have its own value for total CPUs.

Multiple VM Autonomous Exadata VM Cluster is not available on any Oracle Public Cloud deployment of Exadata Infrastructure (EI) resources created before the launch of the Multiple VM Autonomous AI Database feature. For X8M generation and above Exadata Infrastructure resources created after the Multiple AVMC feature launch, each AVMC is created with one cluster node for each of the servers of the Exadata system shape you choose. For information about constraining these total CPUs across different Groups of users, see [How Compartment Quotas Affect CPU Management](#).

Note

The maximum number of AVMC and ACD resources you can create on a given Exadata Infrastructure varies based on the generation of hardware. Please refer to Resource Limits and Characteristics of Infrastructure Shapes for details on constraints for each generation.

At an AVMC or ACD level, the total number of CPUs available for creating databases is called **available CPUs**. At the AVMC resource level, available CPUs will be equal to the total CPUs until you create the first ACD. Once you create an ACD, 8 ECPUs or 2 OCPUs per node are allocated to the new ACD from the AVMC's available CPUs. So, the available CPUs at the

AVMC resource level reduces accordingly. When you create the first Autonomous AI Database in that ACD, the new database consumes the initially allocated CPUs (8 ECPUs or 2 OCPUs per node). If the new database needs more than 8 ECPUs or 2 OCPUs, they get assigned from the parent AVMC's available CPUs, thereby reducing the available CPUs at the parent AVMC level. As you create more ACDs and provision Autonomous AI Databases within each ACD, the available CPU value changes accordingly.

Available CPUs at the Autonomous Exadata VM Cluster level applies to all its Autonomous Container Databases. This count of CPUs available to the container database becomes important if you are using the auto-scaling feature, as described in [CPU Allocation When Auto-Scaling](#).

Similarly, when you manually scale the CPUs of an Autonomous AI Database up, CPUs are consumed from the available CPUs at its parent AVMC level and its value changes accordingly.

When you create an Autonomous AI Database, by default Oracle reserves additional CPUs to ensure that the database can run with at least 50% capacity even in case of any node failures. You can change the percentage of CPUs reserved across nodes to 0% or 25% while provisioning an ACD. See **Node failover reservation** in Create an Autonomous Container Database for instructions. These additional CPUs are not included in the billing.

When an Autonomous AI Database is running, you are billed for the number of CPUs currently allocated to the database, whether specified at initial creation or later by a manual scaling operation. Additionally, if auto-scaling is enabled for the database, you are billed for each second for any additional CPUs the database is using as the result of being automatically scaled up. See CPU Billing Details for more information about how billing is measured and computed.

When an Autonomous AI Database is stopped, you are not billed. However, the number of CPUs allocated to it are not returned to the available CPUs at its parent AVMC level for the overall deployment.

When an Autonomous AI Database is terminated or scaled down, the number of CPUs allocated to it are not immediately returned to the available CPUs at its parent AVMC level for the overall deployment. They continue to be included in the count of CPUs available to its parent container database until that parent container database is restarted. These CPUs are called **reclaimable CPUs**. Reclaimable CPUs at the parent AVMC level is the sum of reclaimable CPUs of all its ACDs. When an ACD is restarted, it returns all its reclaimable CPUs to the available CPUs at its parent AVMC level.

Restarting an Autonomous Container Database (ACD) is an online operation, done in a rolling manner across the cluster, and will not result in application downtime if configured according to best practices to use Transparent Application Continuity.

✓ **Tip**

You can track the different compute (CPU) attributes discussed in this article from the **Details** page of an Autonomous Exadata VM Cluster (AVMC) or Autonomous Container Database (ACD). For guidance, refer to Resource Usage Tracking.

CPU Allocation When Auto-Scaling

The auto-scaling feature enables an Autonomous AI Database to use up to three times more CPU and IO resources than its allocated CPU count. In case of CPU overprovisioning, if three times the CPU count results in a value less than 1, it will be rounded to the next whole number.

CPU overprovisioning is supported with OCPUs only. See CPU Overprovisioning for more details.

To ensure that no single Autonomous AI Database can auto-scale up to consume all CPUs available in the pool for the overall deployment, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure uses the Autonomous Container Database as a limiting control.

While provisioning an auto-scaling enabled Autonomous AI Database in an ACD, if the **available CPUs** in that ACD is less than 3X CPU value of the new database, then additional CPUs will be reserved in that ACD. These CPUs are called **reserved CPUs**. Reserved CPUs ensure that the available CPUs at an ACD level are always greater than or equal to 3x CPU value of the largest auto-scaling enabled database in that ACD. These reserved CPUs can still be used to create or manually scale Autonomous AI Databases in this ACD.

When automatically scaling up an Autonomous AI Database, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure looks for idle CPUs in its parent container database. If idle CPUs are available, the Autonomous AI Database is scaled up; otherwise, it is not. Databases inherently have a lot of idle time, so auto-scaling is a way to maximize resource usage while controlling costs and preserving good isolation from databases in other Autonomous Container Databases.

If the CPU used to auto-scale an Autonomous AI Database came from another running Autonomous AI Database that is lightly loaded and so not is using all of its allocated CPUs, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure automatically scales the auto-scaled database down if the load increases on the other database and it needs its allocated CPU back.

Consider the example of an Autonomous Container Database hosting four running 4-CPU Autonomous AI Databases, all with auto-scaling enabled. The count of CPUs available to the container database for auto-scaling purposes is 12. Should one of these databases need to be auto-scaled past 4 CPUs due to load increase, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure will only perform the auto-scaling operation if one or more of the other databases are lightly loaded and not using all allocated CPUs. The billing cost of this example is 16 CPUs at a minimum because all four 4-CPU databases are always running.

By contrast, consider the example of an Autonomous Container Database hosting four running 2-CPU Autonomous AI Database, all with auto-scaling enabled, and one stopped 8-CPU Autonomous AI Database. The count of CPUs available to the container database for auto-scaling purposes is again 16. Should one of the running databases need to be auto-scaled due to load increase past 2 CPUs, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure can perform the operation using CPUs allocated to the stopped 8-CPU database. In this example, the four running databases can consume up to a total of 8 additional CPUs simultaneously without consuming each other's allocated CPUs. The billing cost of this example is only 8 CPUs at a minimum because only the four 2-CPU databases are always running.

For any Autonomous Data Guard service instance, local or cross-region, the additional pricing will be the number of ECPUs or OCPUs you reserved when you created or explicitly scaled your primary service instance, regardless of whether auto scaling is enabled or not. Auto scaling-related ECPU or OCPU consumption on primary service instances does not occur on Autonomous Data Guard Standby service instances.

How Compartment Quotas Affect CPU Management

Normally, when you create or scale up an Autonomous AI Database, the ability of Oracle Autonomous AI Database on Dedicated Exadata Infrastructure to satisfy your request depends only on the availability of unallocated CPUs in the single pool of CPUs across the entire deployment.

However, you can use the compartment quotas feature of Oracle Cloud Infrastructure to further restrict, on a compartment by compartment basis, the number of CPUs available to create, manually scale and auto-scale Autonomous AI Databases of each workload type (Autonomous AI Lakehouse or Autonomous AI Transaction Processing) individually.

In brief, you use the compartment quotas feature by creating set, unset and zero policy statements to limit the availability of a given resource in a given compartment. For detailed information and instructions, see [Compartment Quotas](#).

How VM Cluster Nodes Affect CPU Management

The preceding discussion of CPU management and allocation states that you can create multiple Autonomous Exadata VM Cluster (AVMC) resources by choosing the CPU count per node while provisioning the AVMC resource.

This section will discuss granular details about how Oracle Cloud Infrastructure places Autonomous AI Databases in the VM cluster nodes, and the consequences of such placement on auto-scaling and parallel processing.

The following attributes determine when and how an Autonomous AI Database is placed across multiple nodes:

- **Split Threshold:** The CPU value beyond which Oracle Cloud Infrastructure opens an Autonomous AI Database across multiple nodes. The default split threshold is 64 for ECPUs and 16 for OCPUs, but if VM Clusters are created with CPU node counts below the default value, then the default is overridden to the VM Cluster node count size. You can also set the split value explicitly using the Split Threshold attribute while provisioning an Autonomous Container Database (ACD).

Autonomous AI Databases created with a CPU value that is smaller than the split value will open on one node in the cluster and those created with a CPU value larger than the split threshold value will open on multiple nodes.

- Suppose you create an ACD with a **default split threshold** (64 ECPUs) in an AVMC with two nodes and 40 ECPUs per node. As 40 is smaller than 64, any Autonomous AI Database with a CPU requirement greater than 40 will be split and opened across multiple nodes, allowing DML requests across those nodes. However, if the AVMC was created with two nodes and 80 ECPUs per node, any database with an ECPU requirement greater than 64 will be split and opened across multiple nodes.
- Suppose you create an ACD in a VM Cluster with two nodes and 40 ECPUs per node and **explicitly set the split threshold value to a much smaller value**, say 20 ECPUs. Any Autonomous AI Database with a CPU requirement greater than 20 will be split and opened across multiple nodes, and databases with a CPU requirement of less than 20 will be opened on a single node.

Setting the split threshold to a much smaller number than the default value increases the chances of databases with *smaller CPU counts* opening on multiple nodes, as long as their CPU count is more than the set split value. Whenever a database is created or scaled to a size greater than this split value, it gets opened on multiple nodes. This is useful when you want databases to open on multiple nodes to control performance degradation in case of a node failure or planned maintenance. With databases split across multiple nodes in larger RAC clusters, if any one node fails or when scheduled maintenance occurs, you can continue to have higher performance rather than degrading to a 50% performance profile.

- Suppose you **explicitly set the split threshold to a value much higher** than default, say 80 ECPUs, in an AVMC with two nodes and 40 ECPUs. Any Autonomous AI Database with a CPU requirement greater than 40 will be split and opened across

multiple nodes, and databases with a CPU requirement of less than 40 will be opened on a single node.

Setting the split threshold to a value much higher than the default causes your database DML to stay on a single RAC node and eliminate the chance of cluster wait contention.

- **When you manually scale an Autonomous AI Database**, the new CPU value will be applied to the existing Split model. That is, if the new value is smaller than the split threshold, it will open on one node, and if the value is greater than the split threshold, it will open on multiple nodes.
- **Distribution Affinity:** Determines the number of nodes on which an Autonomous AI Database will be opened once it crosses the Split Threshold.

For example, suppose you created an AVMC resource with 4 nodes and 80 ECPUs per node, and you created an ACD in this AVMC with the database **Split Threshold** set to 64. Creating an Autonomous AI Database with an ECPU requirement of 120 will split and open the database across multiple nodes as 120 greater than 64 (Split Threshold).

- If your distribution affinity is set to **Minimum nodes**, Oracle Cloud Infrastructure tries to create the database on 2 nodes with 60 ECPUs on each node. If this is not possible, it will be split across 3 nodes with 40 ECPUs each. If that is also not possible, then Oracle Cloud Infrastructure will try to open the database across 4 nodes with 30 ECPUs each.
- If you specify distribution affinity to **Maximum nodes**, Oracle Cloud Infrastructure tries to create the database split across all 4 nodes with 30 ECPUs each. If this is not possible, it will be split across three nodes with 40 ECPUs each. If that is also not possible, then Oracle Cloud Infrastructure will try to open the database across 2 nodes with 60 ECPUs each.
- **Node Failover Reservation (%):** The number of CPUs set aside on adjacent nodes (nodes where your database software is present but not open) in your AVMC for localized failure and maintenance events. Node Failover Reservation applies to non-Split database deployments. By default, there is a 50% reserve, meaning during a failure event or maintenance, you will continue to run but at 50% of the allocated CPU.
 - For non-critical databases or databases with very light utilization, you can set Node Failover Reservation to a smaller value so that, in the end, you can create and consolidate a larger number of databases on your Dedicated Exadata Infrastructure.
 - You can set this value to zero for development environment and databases where a downtime during maintenance is acceptable.
 - To some extent, Node Failover Reservation can also be controlled by ensuring a database is split across more than 2 nodes, using split threshold and distribution affinity. Consider a scenario where an Autonomous AI Database is split across 4 nodes. By removing one node at a time in a rolling fashion while a maintenance activity is in progress, you always have 3 nodes still up and taking traffic, keeping your performance reserve effectively 75%, rather than the usual 50%. With larger clusters, you can drive this reserve up even further, say to an 87.5% reserve on an 8-node cluster.

How an Autonomous AI Database's CPU allocation is distributed across VM cluster nodes affects the following operations:

- **Auto-scaling:**
 - Auto-scaling may occur within a single VM cluster node for non-parallelizable DML and across VM Cluster nodes if the DML is parallelizable.

- Multiple concurrent session with non-parallelizable queries may be routed to all nodes in the cluster, effectively allowing auto-scaling across all nodes in a multi-node database.
- **Parallel Processing:**
 - Parallel processing of SQL statements occurs within Autonomous Exadata VM cluster nodes that are open, first within a single node, and then in adjacent open nodes, which as discussed above will depend on the size of the Autonomous Exadata VM Cluster.

Based on the resource utilization on each node; not all the values of the available CPUs can be used to provision or scale Autonomous AI Databases. For example, suppose you have 20 CPUs available at the AVMC level, not all the values from 1 to 20 CPUs can be used to provision or scale Autonomous AI Databases depending on the resource availability at the node level. The list of CPU values that can be used to provision or scale an Autonomous AI Database is called **provisionable CPUs**.

When you try to provision or scale an Autonomous AI Database from the OCI console, the CPU field will give you a dropdown with the list of provisionable CPUs. Alternatively, you can use the following APIs to get the list of provisional CPU values:

- **GetAutonomousContainerDatabase** returns a list of provisionable CPU values that can be used to create a new Autonomous AI Database in the given Autonomous Container Database. See [GetAutonomousContainerDatabase](#) for more details.
- **GetAutonomousDatabase** returns a list of provisionable CPU values that can be used for scaling a given Autonomous AI Database. See [GetAutonomousDatabase](#) for more details.

Consolidate Autonomous AI Database Instances Using Elastic Pools

Use an elastic pool to consolidate your Autonomous AI Database instances, in terms of their allocation of compute resources, and to provide up to 87% cost savings.

Elastic pools help you improve operating efficiency and reduce costs by bringing all of your databases to the Cloud. This also supports consolidating resources and simplifying administration and operations by using Autonomous AI Database. When you need a large number of databases, that can scale up and down elastically without downtime, you can benefit by creating and using elastic pools.

When you create an elastic pool you select a pool size from a predefined set of pool sizes. Pool size determines how much you pay for compute as well as how many ECPUs you can provision in a given pool.

Note

Elastic pools are only available for Autonomous AI Database instances that use the ECPU compute model without Autonomous Data Guard.

About Elastic Pools

There are several terms to use when you work with elastic pools:

- **Pool Leader:** The Autonomous AI Database instance that creates an elastic pool.
- **Pool Member:** An Autonomous AI Database instance that is added to an elastic pool.
- **Pool Size:** A value you set when creating an elastic pool. The pool size must be one of the available elastic pool shapes.

- **Pool Shape:** When you create an elastic pool, you select a pool shape from among the valid pool sizes. The shape must have one of 128, 256, 512, 1024, 2048, or 4096 ECPU.
- **Pool Capacity:** The maximum number of ECPU that an elastic pool can use. It is four times (x4) the pool size.

The following apply with elastic pools:

- Provisioning a pool leader or a member is subject to service limits enforced at the tenancy or compartment levels.
- Starting and stopping a pool member database does not depend on the pool leader's state. You can independently stop and start the databases part of an elastic pool, including the pool leader and members.
- In an elastic pool, the pool leader's licensing selections determine the license requirements for the entire pool. That is, as long as an Autonomous AI Database is a member of an elastic pool, its license selections do not apply, and they come into effect only after it leaves the elastic pool.

Benefits of Using Elastic Pools

Elastic pools provide the following benefits. They:

- Enable operating with a fixed budget for a group of databases while delivering performance elasticity for each individual database.
- Allow for easy migration from on-premise Oracle environments that include oversubscription to provide a cost-effective way to move to an Autonomous AI Database.
- Support SaaS vendors with a large number of individual customer databases.
- Provide resources for using a microservices architecture, where the ability to supply a large number of databases is required.
- The pool members in an elastic pool are not billed individually (the pool leader is billed based on the pool shape). You can allocate additional ECPU per instance for pool members without worrying about the cost associated with the ECPU usage for the individual members.
- Autonomous AI Database memory allocation is directly correlated with the ECPU count, so selecting a greater number of ECPU for instance allows you to run with more memory without paying for the additional resources. This means using a larger number of ECPU per instance will enable you to use more memory per instance, where the cost is based on the pool shape and not on an individual instance's ECPU count.

Requirements to Create an Elastic Pool

The following are the requirements for an Autonomous AI Database instance to create an elastic pool and become a pool leader:

- The instance must use the ECPU compute model without Autonomous Data Guard.
- The instance must be an Autonomous AI Database with Autonomous AI Transaction Processing workload type. This only applies to the pool leader. An elastic pool can hold a mix of databases with Autonomous AI Transaction Processing and Autonomous AI Lakehouse workloads.
- Auto scaling must be disabled.
- The instance must not be a member of an existing elastic pool.

- The maximum allowed individual ECPU count for an Autonomous AI Database instance that creates an elastic pool is 4 times the pool size specified when you create the pool.
- The instance that creates an elastic pool is subject to tenancy limits. To create an elastic pool, you must have a sufficient number of ECPUs available, below the tenancy limit, to accommodate the size of the elastic pool.

Requirements to Join an Elastic Pool

The following are the requirements for an Autonomous AI Database instance to join an elastic pool:

- The instance must:
 - Use the ECPU compute model without Autonomous Data Guard.
 - Be in the same tenancy and region as the pool leader.
 - Not be a member of an existing elastic pool.
 - Have auto-scaling disabled.
- The instance can be from a different AVMC or a different Exadata Infrastructure to the leader.
- An elastic pool can contain Autonomous AI Database instances with Autonomous AI Transaction Processing and Autonomous AI Lakehouse workload types.
- The available pool capacity is the maximum allowed individual ECPU count for an Autonomous AI Database instance. When an instance's ECPU count is greater than the available pool capacity, it is not allowed to join that elastic pool.

Pool Leader and Member Instance ECPU Allocation

When an Autonomous AI Database instance is part of an elastic pool:

- The minimum allowed individual ECPU allocation for an instance is 1 ECPU.
- Increments of 1 ECPU are allowed for individual Autonomous AI Database instance ECPU allocation.

Pool Capacity for an Elastic Pool

An elastic pool has a pool capacity of 4 times the pool size. For example, a pool with pool size of 128 ECPUs can hold up to 512 ECPUs for its leader and the members.

The following are examples of Autonomous AI Database instances that could be in an elastic pool with a pool size of 128 and a pool capacity of 512 ECPUs:

- Each of these are valid for pool members in an elastic pool with a pool size of 128 ECPUs:
 - 1 instance with 512 ECPUs, for a total of 512 ECPUs
 - 128 instances with 4 ECPUs, for a total of 512 ECPUs
 - 256 instances with 2 ECPUs, for a total of 512 ECPUs
- Similarly, each of the following are valid for pool members in an elastic pool with a pool size of 128 ECPUs:
 - 1 instance with 128 ECPUs, 2 instances with 64 ECPUs, 32 instances with 4 ECPUs, and 64 instances with 2 ECPUs, for a total of 512 ECPUs

- 256 instances with 1 ECPU, 64 instances with 2 ECPUs, for a total of 384 ECPUs, which is less than the pool capacity of 512 ECPUs.

Elastic Pool Leader Operations

The Autonomous AI Database instance that creates an elastic pool is the pool leader.

The following operations are valid only for the pool leader:

Operation	Description
Create an elastic pool.	The Autonomous AI Database instance that creates an elastic pool is the pool leader. See Create an Elastic Pool for more information.
Remove an elastic pool member.	An elastic pool leader can remove a member from the elastic pool. See Remove Pool Members from an Elastic Pool for more information.
Terminate an elastic pool.	When an elastic pool has no pool members, the pool leader can terminate the elastic pool. See Terminate an Elastic Pool for more information.
Modify elastic pool size.	An elastic pool leader can modify the pool size. See Change the Elastic Pool Shape for more information.
List elastic pool members.	A pool leader can list pool members. See List Elastic Pool Members for more information.

Elastic Pool Member Operations

The Autonomous AI Database instance that are added to an existing pool are pool members.

The following operations are valid only for a pool member or a pool leader:

Operation	Description
Join an elastic pool.	You can add an Autonomous AI Database instance to an elastic pool as a member <i>only when</i>: • There are sufficient ECPUs remaining in the pool for allocation (4x the Pool Size). • The instance is one of the supported workload types, that is, Transaction AI Processing or Lakehouse. • Its compute model is ECPU. • It is not already a member of a different pool. • It is not configured with Autonomous Data Guard. See Join an Elastic Pool and Join an Elastic Pool While Provisioning or Cloning an Autonomous AI Database for guidance.

Operation	Description
Leave an elastic pool.	As a pool member, you can remove your instance from an elastic pool. - When a pool member leaves an elastic pool: - Auto-scaling is disabled. After leaving the elastic pool, you can enable auto-scaling for the instance. - The pool will have more resources available. For example, if the elastic pool were fully allocated up to the pool capacity, and an instance with 10 ECPUs leaves the pool, the elastic pool would have 10 available ECPUs. See Leave an Elastic Pool Autonomous AI Database for instructions.

Change Management

Service Maintenance of Autonomous AI Database on Dedicated Exadata Infrastructure

Oracle schedules and performs all patching and other maintenance operations on all the Autonomous AI Database resources of on Dedicated Exadata Infrastructure. At the same time, it provides you with various options to customize, view, and reschedule maintenance events for the different infrastructure resources.

① Note

With Database In-Memory enabled, you may experience performance degradation during any patching activity that results in rebooting the database. For additional information on Database In-Memory, see [Database In-Memory](#).

Service Maintenance Types

Oracle schedules and performs different service maintenance activities on your Autonomous AI Database. These maintenance events vary in their scope and frequency of patching.

Oracle's Cloud Operations team continuously monitors patching and will perform a rollback if a patch fails basic sanity tests. If a rollback is required, the maintenance will be rescheduled. While rollback is the last option, our goal is always to provide the fastest remediation to restore your database to a healthy state. If a regression appears only within your application, it should be reported via a service request (SR). For critical issues requiring immediate attention, Oracle can develop and deploy a one-off patch outside the standard maintenance schedule.

- Quarterly Maintenance Patches:** In general, Oracle schedules and performs entire fleet maintenance spread throughout *each quarter*.
 - Quarterly maintenance patches are applied at various resource levels such as Exadata Infrastructure, Autonomous Exadata VM Cluster (AVMC), and Autonomous Container Database (ACD). The quarterly maintenance window can be set while creating these infrastructure resources or changed later.

- You can let Oracle handle maintenance scheduling, or you can set a specific maintenance window when Oracle can begin maintenance operations.
- By default, Oracle applies Release Updates (RUs) along with these quarterly maintenance patches. You can configure to update RU in a Rolling or Non-Rolling maintenance method.
 - * The **Rolling** method updates the ACD, one node at a time, with no downtime for the Autonomous AI Databases.
 - * The **Non-rolling** method shuts down and updates the ACD in parallel across all nodes. This method minimizes maintenance time but it requires full downtime for the ACD and all associated Autonomous AI Databases.

Note

In an Autonomous Data Guard configuration, the non-rolling maintenance method results in downtime for the primary and standby ACDs during their respective maintenance window until the patching completes.

- You can also include the time-zone file to be updated along with the RU. Quarterly maintenance patches that include a time-zone file update would require a complete downtime for the ACD and the associated Autonomous AI Databases. The downtime is dependent on the amount of data that is time-zone sensitive.
- Quarterly maintenance patches without a time-zone file update can be applied in a rolling or non-rolling method, depending on the maintenance configuration of the Autonomous Container Database (ACD).
- **Monthly Security Patches:**
 - **Exadata Infrastructure Security Patches:** Oracle schedules and performs a *monthly infrastructure security maintenance activity* alongside quarterly maintenance. However, these security patches are applied only in those months with critical security updates, including fixes for vulnerabilities with CVSS scores greater than or equal to 7.
 - * Any Exadata Infrastructure provisioned before Oracle schedules the security maintenance, will be eligible for security maintenance.
 - * The monthly security maintenance process updates database servers to fix critical security vulnerabilities and product issues. They also update storage servers to an Exadata Storage Software image that resolves known security vulnerabilities and product issues.
 - **Autonomous VM Cluster Security Patches:** Oracle performs monthly security maintenance for Autonomous VM Clusters in addition to the regular quarterly updates. These patches are only applicable for GOV regions.
 - * Monthly security patches are applied using the rolling method.
 - * The first month of each quarter includes quarterly patches; the following two months include monthly security patches.
 - * To ensure patches are applied, you must select all three months of the quarter and specify a preference for Week 3 and/or Week 4.
- **One-off Patches:** Oracle generates one-off patches for critical support requests filed with My Oracle Support. For help with filing a support request, see [Create a Service Request in My Oracle Support](#).

- When you and Oracle agree that a service request is critical and requires a one-off patch for immediate resolution, the service team generates a one-off patch and makes it available. One-off patches are separate from scheduled maintenance patches.
- If you enable Oracle Cloud notifications and events with a rule to receive notifications regarding new updates, then when one-off patches become available, Oracle sends a notification that contains the OCID of the product to be patched. Otherwise, you can find the update availability notice in the My Oracle Support portal for the support request that you filed.
- The one-off patches are forward merged into the next Release Update (RU) to ensure that:
 - * A one-off fix delivered for a particular customer is available to all the customers.
 - * The subsequent versions do not need to apply one-off patches again.
- If required, an RU can have multiple one-off patches merged with it. As of the current release, the one-off patches are not cumulative, and therefore, you must apply them individually. If a one-off patch is too close to the subsequent RU, a custom version of the RU with the one-off fix is created for the next quarter.
- Suppose a one-off fix that is not merged into the latest RU is scheduled, and you choose to apply the next RU. Then, Oracle cancels the scheduled one-off patch. You can view the canceled maintenance runs in the maintenance history. All the one-off patching details logged into the maintenance history are available in downloads and audit and logging services.
- If needed, a one-off patch can be rolled back via a service request.
- The number of one-off patches available for an Autonomous Container Database is shown on its **Details** page. Clicking the **Copy** link next to it copies all those one-off patch numbers.

Specify When Maintenance Can Occur

In general, Oracle schedules and performs entire fleet maintenance spread throughout each quarter and monthly infrastructure security fixes for vulnerabilities with CVSS scores greater than or equal to 7. You can let Oracle handle maintenance scheduling, or you can set a specific maintenance window when Oracle can begin maintenance operations.

Customizing Quarterly Maintenance

You either choose a schedule for quarterly automatic maintenance of the Autonomous AI Database resources or let Oracle schedule the updates automatically. In advance, Oracle notifies you of the date and time of upcoming scheduled maintenance.

You can perform the following with automatic quarterly maintenance at various resource levels, as listed in the table below:

- Customize the automatic maintenance preferences and schedule. You can set these preferences while provisioning the Autonomous AI Database resources or change them later.
- View and change the schedule at any point before the beginning of the scheduled maintenance. Changes made to the scheduled maintenance for subsequent quarters do not affect the schedule for the current quarter.
- View the past maintenance events.

Infrastructure Resource	Notes & Further Reference
Exadata Infrastructure (EI)	• Change the Maintenance Schedule of an Exadata Infrastructure Resource. There are various options to customize the maintenance schedule as explained in Settings in Maintenance Schedule that are Customizable. • View and Manage Scheduled Maintenance of an Exadata Infrastructure Resource • View Past Maintenance of an Exadata Infrastructure Resource
Autonomous Exadata VM Cluster (AVMC)	• Change the Maintenance Schedule of an Autonomous Exadata VM Cluster. There are various options to customize the maintenance schedule as explained in Settings in Maintenance Schedule that are Customizable. • View and Manage Scheduled Maintenance of an Autonomous Exadata VM Cluster • View Past Maintenance of an Autonomous Exadata VM Cluster

Note

AVMC resources provisioned on the Exadata Infrastructure resources in Oracle Cloud before the launch of the multiple VM Autonomous AI Database feature inherit the maintenance schedule from its associated Exadata Infrastructure.

Infrastructure Resource	Notes & Further Reference
Autonomous Container Database (ACD)	- Update Autonomous Container Database Maintenance Preferences demonstrates how to update the following preferences: Maintenance method for the automatic updates (Rolling or Non-rolling). You can also include the time-zone file to be updated along with the RU. Note In an Autonomous Data Guard configuration, the non-rolling maintenance method results in downtime for the primary and standby ACDs during their respective maintenance window until the patching completes. Maintenance version for the automatic updates (Next RU or Latest RU). Automatic maintenance schedule for the ACD. There are various options to customize the maintenance schedule as explained in Settings in Maintenance Schedule that are Customizable. Note You can not define a custom schedule for a standby ACD in an Autonomous Data Guard configuration. However, you can set number of days for which the standby ACD maintenance will be scheduled before primary ACD maintenance because standby ACD is always patched before primary ACD. - You can also schedule an on-demand maintenance to update RU (Release Update) along with the time-zone file or just the time-zone file for an ACD. Refer to Schedule a Quarterly Maintenance Update for instructions. Note Only in case of on-demand time-zone file updates, the standby ACD will be patched 3 days before the primary ACD in Autonomous Data Guard enabled ACDs. - View and Manage Scheduled Maintenance of an Autonomous Container Database

Infrastructure Resource	Notes & Further Reference
	View Past Maintenance of an Autonomous Container Database

✓ Tip

Oracle recommends that you set a maintenance window for all the infrastructure resources listed above to:

- Prevent maintenance operations from occurring at times that would be disruptive to regular database operations.
- Patch your infrastructure resources in a staggered manner. Staggering the maintenance events for different infrastructure resources is a best practice and can help you verify the patch on a set of resources before patching another. For example, when you are using different Autonomous Container Databases for development, and testing and you want to verify the patches on your development environment before applying them to production, you can customize their maintenance schedules so that all your development ACDs are patched before the production ACDs.

Settings in Maintenance Schedule that are Customizable

You can choose the following details from the Oracle Cloud Infrastructure console while defining a custom schedule for any of the above infrastructure resources.

- Permitted Months:** You must select at least one month per quarter, and you can also choose to skip patching for a quarter. Patching cannot be skipped for two consecutive quarters.

📘 Note

When you choose to skip, you need to select at least one month from that quarter. This acts as a fallback in case maintenance did not occur in the previous unskipped quarter. In this scenario, Oracle will automatically perform the maintenance in the selected month, even if skipping is chosen for that quarter.

- Week (or Weeks) Within the Selected Months:** Weeks start on the 1st, 8th, 15th, and 22nd days of the month and have a duration of 7 days. Weeks start and end based on calendar dates, not days of the week. Maintenance cannot be scheduled for the fifth week of months that contain more than 28 days. If you do not specify a week of the month, Oracle will assign a week automatically.
- Day (or Days) of the Selected Week:**

If you do not specify a day of the week, Oracle will run the maintenance update on a day that it assigns automatically.

As the weeks begin and end based on calendar dates and not on days of the week, you must pay extra attention while choosing the days if you want to ensure a specific sequence in patching your Exadata Infrastructure (EI). For example, observe the two months shown below:

September, 2023

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

October, 2023

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

For September 2023, week 1 begins on a Friday and ends on a Thursday. So, its first Saturday comes *a day before* its first Sunday. However, week 1 in October 2023 begins on a Sunday and ends on a Saturday. As a result, its first Saturday *comes five days after* its first Sunday.

Suppose you wish to patch all the Exadata Infrastructure resource before patching its Autonomous Container Databases (ACDs) to maintain a specific sequence for maintenance. Scheduling the maintenance of the Exadata Infrastructure resource on week 1 Saturday and its ACDs on week 1 Sunday assuming that week 1 Sunday always comes a day after week 1 Saturday may work for a few months such as *September 2023* but not others like *October 2023*. If you want to implement a specific sequence for patching, it might be a better option to *space them by a week*. In this case, you can schedule your Exadata Infrastructure resource on week 1 Saturday and its ACDs on week 2 Sunday. Doing so will always ensure that your Exadata Infrastructure resource is patched before patching its ACDs.

- **4-Hour Window (or Windows)** when maintenance operations can begin.
- **Buffer Period Between Primary and Standby Maintenance:** Number of days between the standby ACD maintenance and the primary ACD maintenance, that is, how many days before maintenance on the primary container database is performed that maintenance on the standby container database is to be performed. You can choose any value from 1 to 7 days.
 - ✓ Selecting the buffer period is applicable only to an Autonomous Container Database that is the primary database in an Autonomous Data Guard configuration.
- **Lead Time:** The minimum number of weeks ahead of the maintenance event you would like to receive a notification message. Your lead time ensures that a newly released maintenance update is scheduled to account for your required minimum period of advanced notification.
 - ⊗ Lead Time is not applicable for the maintenance of Autonomous Container Database resources.
- You can revert changes back to the default settings by selecting **Reset to default**.

Customizing Monthly Infrastructure Security Maintenance

Monthly infrastructure security maintenance, when needed, is scheduled to be applied during a 21-day window that begins between the 18th and 21st of each month and will run till the 9th to the 12th of the following month. You will receive notification of the proposed schedule at least 7 days before the start of the monthly maintenance window, and you can reschedule monthly maintenance to another date in the window if desired.

Monthly security patches can be rescheduled to another time within the maintenance window but cannot be skipped or rescheduled beyond the 21-day window. You can reschedule monthly security maintenance when rescheduling the quarterly maintenance as long as you keep the monthly within the current maintenance window.

There is no impact on the Autonomous AI Databases or applications connected to them during the monthly Infrastructure security patching activity. Updates to database servers are applied online via Ksplice technology, and updates to storage servers are applied in a rolling fashion.

However, while updating your service infrastructure, Oracle may block some operations, including memory, and storage scaling, operating system and Grid Infrastructure patching (including prechecks), and elastic expansion of compute and storage servers. Please plan to defer these operations until after the updates are complete. Application of security updates takes about 15 minutes per DB server host, plus 60 minutes per storage server, depending on the I/O activity. If you attempt an affected operation, the console will notify you of the ongoing security updates. No software is updated in the guest VMs.

Customizing One-off Patches

Using the Oracle Cloud Console Maintenance View, you can edit the scheduled start time or choose to install the one-off patch immediately. By default, Oracle schedules a one-off patch to be applied within 72 hours of the patch becoming available. If no action to change the schedule occurs, then the patch is automatically applied. You can reschedule the one-off patches only within the current quarter. However, you can not skip a one-off patch altogether.

Specify What Kind of Patches Apply

One standard maintenance operation is to apply database software patches to your Autonomous Container Databases and, by extension, the Autonomous AI Databases created in them. By default, Oracle applies Release Updates (RUs). You can configure the maintenance type to either Next RU to update the Autonomous Container Database to the next release update or the Latest RU to update Autonomous Container Database to the latest release update in the next maintenance window. Accordingly, Oracle will use an image type that meets your preference when available. You always have the option to change a given scheduled patch to a different version when desired.

For step-by-step guidance, see [Update Autonomous Container Database Maintenance Preferences](#).

View and Manage Already Scheduled Maintenance

Once a maintenance activity is scheduled based on the maintenance window you set, you can manage the actual timing of the activity, even to the point of changing the patch version, applying the patch immediately, or skipping the activity.

Scheduled Maintenance Details

For each scheduled Exadata Infrastructure, Autonomous Exadata VM Cluster, or Autonomous Container Database maintenance event, the resource's Maintenance page lists the following details:

- The status of the event
- The type of the event, as Weekly, Quarterly, Monthly, or Yearly
- The OCID of the event
- The scheduled start time and date of the event
- The maintenance method of the event as Rolling on Non-Rolling. This is displayed *only* for an Exadata Infrastructure resource.
- The patch version to be applied in the event. This is displayed *only* for an Autonomous Container Database resource.

Management Operations on a Scheduled Maintenance

For each maintenance event listed in an infrastructure resource maintenance page, you can perform the following management operations, provided that the event is not already in progress:

- Reschedule the start time and date of the event to a later time in the quarter. Specify the new start time and date in the **Edit Maintenance Start Time** window.
- Start the maintenance event immediately by clicking **Patch Now**.

Note

Patch Now is not available for an Autonomous AI Database enabled with Autonomous Data Guard. As a workaround, you can modify the scheduled maintenance time to begin in the nearest 4 hour period available. Ensure that the standby is patched before the primary with a buffer period of 1 to 7 days between them.

- Skip an *Autonomous Container Database* scheduled maintenance event.

Note

You can not skip two consecutive maintenance events. After skipping a maintenance event, you can not skip the next *immediate* scheduled maintenance event; you can only skip the maintenance events for two alternate quarters in a year.

- Select a different patch version to apply. When selecting a version, note the following:
 - You must select a version that is later than the current version of the Autonomous Container Database.
 - The list of available versions may contain both release updates (RUs) and release update revision (RURs). You can choose either type, regardless of the maintenance type configured for the Autonomous Container Database. Selecting a different type in the **Version** list does not change the type configured for the Autonomous Container Database.

- Update the *Exadata Infrastructure* maintenance method from Rolling to Non-rolling and vice-versa.

For step-by-step guidance, see:

- View and Manage Scheduled Maintenance of an Exadata Infrastructure Resource
- View and Manage Scheduled Maintenance of an Autonomous Exadata VM Cluster
- View and Manage Scheduled Maintenance of an Autonomous Container Database

View Maintenance Status Notifications

The `DB_NOTIFICATIONS` view stores information about maintenance status notifications for your Autonomous AI Database instance.

APPLIES TO:  Oracle Public Cloud only

To show notification information:

1. Connect to your Autonomous AI Database instance.
2. Use the following query to view maintenance (patching) information.

```
SQL> SELECT * FROM DB_NOTIFICATIONS WHERE TYPE = 'MAINTENANCE';
```

The following provides details about the maintenance status.

- Maintenance run has ended: Specifies maintenance has completed. The STATUS shows the value COMPLETED with the start and end timestamps for the completed maintenance in ACTUAL_START_DATE and ACTUAL_END_DATE.
- Maintenance run is scheduled for the instance: Specifies a new maintenance has been scheduled. The STATUS shows the value SCHEDULED with the expected start and end timestamps for the scheduled maintenance in EXPECTED_START_DATE and EXPECTED_END_DATE.
- Maintenance run has begun: Specifies the maintenance is in progress and provides the start timestamp for the active maintenance. The STATUS shows the value IN_PROGRESS and ACTUAL_START_DATE stores the start timestamp.

Table 3-4 The following table shows the DB_NOTIFICATIONS columns and datatypes.

Column	Datatype	Description
TYPE	VARCHAR2(128)TYPE	Specifies the type of the notification. Valid value is: MAINTENANCE.
TIME	TIMESTAMP(6) WITH TIME ZONE	Time when the notification entry was added.
EXPECTED_START_DATE	TIMESTAMP(6) WITH TIME ZONE	Scheduled maintenance start time.
EXPECTED_END_DATE	TIMESTAMP(6) WITH TIME ZONE	Scheduled maintenance endtime.
ACTUAL_START_DATE	TIMESTAMP(6) WITH TIME ZONE	Actual maintenance start time.
ACTUAL_END_DATE	TIMESTAMP(6) WITH TIME ZONE	Actual maintenance end time.

Table 3-4 (Cont.) The following table shows the DB_NOTIFICATIONS columns and datatypes.

Column	Datatype	Description
PRODUCT	VARCHAR2(128)	Product/component for which maintenance is scheduled/ongoing. Values: DATABASE, GI, DOM0, DOM0_CELL, DOMU, CELL, IBSWITCH.
STATUS	VARCHAR2(128)	Current status of the maintenance. Values: SCHEDULED, IN_PROGRESS, COMPLETED, FAILED, SKIPPED.
OP_MODE	VARCHAR2(64)	Patching operation mode. Values: RAC_ROLLING, NON_ROLLING.
DATABASE_IMPACT	VARCHAR2(64)	Database impact. Values: AVAILABILITY, PERFORMANCE, NONE.
DESCRIPTION	VARCHAR2(128)	The notification message details.
PATCH_ID	VARCHAR2(128)	Patch version.

Automatic Queuing of Maintenance Events

Quarterly Maintenance Events of Different Autonomous AI Database Resources

If you choose a custom maintenance schedule for any infrastructure resources, Oracle honors your preference while scheduling the maintenance events. However, should your custom schedule create any overlap with other infrastructure resources, Oracle automatically serializes such that the maintenance events execute in this sequence with some time gap between them; Exadata Infrastructure, Autonomous Exadata VM Cluster, Autonomous Container Database.

Example: Suppose an Exadata Infrastructure resource maintenance event and an Autonomous Container Database maintenance event are scheduled to start simultaneously. In that case, the Exadata Infrastructure resource maintenance event starts, and the Autonomous Container Database maintenance event is queued and begins immediately after the Exadata Infrastructure resource maintenance event.

Quarterly Maintenance Events and Monthly Infrastructure Security Patches

Scenario	Queuing
When a quarterly maintenance activity is scheduled within 24hours of a monthly infrastructure security patch.	The scheduled monthly maintenance will be skipped, and applied immediately following the quarterly maintenance.
When a quarterly maintenance activity is scheduled at the same time as a monthly infrastructure security patch.	The quarterly maintenance will be performed first, and the monthly security patch will be applied immediately following the completion of the quarterly maintenance.

Scenario	Queuing
When a monthly infrastructure security patch is scheduled to begin 0-24 hours ahead of the quarterly maintenance.	The scheduled monthly maintenance will wait and will be performed immediately following the quarterly maintenance. If the quarterly maintenance is subsequently rescheduled, then the monthly security maintenance will begin immediately. Oracle, therefore, recommends scheduling quarterly and monthly maintenance at the same time. As a result, if you reschedule the quarterly maintenance event at the last moment, the monthly maintenance activity will run at the scheduled time upon editing the schedule.
When a quarterly maintenance is scheduled outside the 24-hour window of the security maintenance in the same month.	You will need one maintenance window for quarterly maintenance and one maintenance window for security maintenance.

Note

Any time before the scheduled monthly Exadata Infrastructure maintenance, you can reschedule it.

The storage servers will only be updated once if you schedule the monthly security maintenance **at least 25 hours before** the quarterly maintenance in the month where both quarterly and monthly security maintenance has been scheduled.

View Past Maintenance Events

You can view past maintenance of an Exadata Infrastructure, Autonomous Exadata VM Cluster, or an Autonomous Container Database resource from its **Details** page.

For step-by-step guidance, see:

- View Past Maintenance of an Exadata Infrastructure Resource
- View Past Maintenance of an Autonomous Exadata VM Cluster
- View Past Maintenance of an Autonomous Container Database

Monitor Service Maintenance Events

You can monitor your Autonomous AI Database infrastructure resources' maintenance events using the Events and Notifications services. Using the Events and Notifications services, you can get email notifications when maintenance events occur on Exadata Infrastructure, Autonomous Exadata VM Cluster, and Autonomous Container Database resources.

For each infrastructure resource, four different maintenance events are generated as listed below:

- Maintenance Scheduled

- **Maintenance Reminder**
For the Autonomous Exadata VM Cluster (AVMC) and Autonomous Container Database (ACD) resources, the maintenance reminder notification is sent 1 week before the actual maintenance run. For Exadata Infrastructure resources, the reminder notification is released between 1 to 4 weeks before the maintenance run, depending on the set preference.
- **Maintenance Begin**
- **Maintenance End**

For the complete list of events generated for each infrastructure resource, see [Events for Autonomous AI Database on Dedicated Exadata Infrastructure](#).

You can subscribe to any of these maintenance events for an infrastructure resource by performing the following high-level tasks:

- Create a notifications service topic.
- Add an email subscription to the topic.
- Add an events service rule to send maintenance events to the notifications service topic.

Example 3-1 Notifications: Emails for Maintenance Events

For a step-by-step guide with an example, see [Notifications Example: Emails for Maintenance Events](#).

Observability

Plan and Observe Capacity

You can observe and plan the compute and storage resources of your Autonomous AI Database on Dedicated Exadata Infrastructure for efficient usage and optimal billing.

Oracle Autonomous AI Database provides dashboards and visualizations to help you track resource allocation and usage for your service.

Resource Terminology

It is important to understand the various terms used with resource allocation and usage on the Oracle Cloud Infrastructure (OCI) console and understand what they mean:

- **Available CPUs:** CPUs available for allocation to provision new Autonomous AI Databases or scale existing Autonomous AI Databases.
- **Provisioned CPUs:** Total CPUs allocated for all the Autonomous AI Databases within an Autonomous Exadata VM Cluster (AVMC).
- **Reserved CPUs:** Total CPUs reserved to support auto-scaling of Autonomous AI Databases, Autonomous AI Database failover on node failure, and life-cycle management of empty Autonomous Container Databases (ACDs).
- **Reclaimable CPUs:** Total CPUs from all terminated and scaled down Autonomous AI Databases in all the Autonomous Container Databases (ACDs) within an Autonomous Exadata VM Cluster (AVMC). Reclaimable CPUs are not returned to the Available state until the ACD is restarted.

 Tip

See Compute Management in Autonomous AI Database for a detailed explanation about the CPU types with examples.

- **Provisionable ACDs:** Number of Autonomous Container Databases that can be created in an Autonomous Exadata VM Cluster (AVMC).
- **Provisioned ACDs:** Number of Autonomous Container Databases that have been created in an Autonomous Exadata VM Cluster (AVMC).
- **Non-provisionable ACDs:** Number of Autonomous Container Databases that cannot be created because of a shortage of available CPUs in an Autonomous Exadata VM Cluster (AVMC).
- **Total Storage (in TBs):** Total storage allocated to an Autonomous Exadata VM Cluster (AVMC).
- **Available Storage (in TBs):** Storage available for Autonomous AI Databases' use in an Autonomous Exadata VM Cluster (AVMC).
- **Used Storage (in TBs):** Storage currently used by Autonomous AI Database(s) in an Autonomous Exadata VM Cluster (AVMC).
- **Memory per CPU (in GBs):** Memory allocated to the Autonomous Exadata VM Cluster (AVMC) per CPU.

Resource Limits

The following table lists the resource limits for Autonomous AI Database on Dedicated Exadata Infrastructure deployments on Oracle Public Cloud and Exadata Cloud@Customer.

Resource Limits (Maximum)

- The number of Available CPUs determines the maximum number of Autonomous AI Databases that you can create. The smallest Autonomous AI Database can be created with 2 ECPUs or 0.1 CPUs. Therefore the maximum number of **Autonomous AI Databases** that can be created is *half* times the amount of Available ECPUs or *ten times* the amount of Available OCPUs.
- The maximum number of **Autonomous Container Databases per Autonomous Exadata VM Cluster (AVMC)** is 16.

Recommended Resource Limits (Maximum)

Resource	Recommended Limit
Autonomous AI Databases per Autonomous Container Database	200
Autonomous AI Databases per Autonomous Container Database with Autonomous Data Guard Configured	25

Note

It is possible to provision more Autonomous AI Databases than those mentioned in the above recommended limits table, especially with CPU overprovisioning. However, this implies *compromising* the Service Level Objectives (SLOs) to return an application online following an unplanned outage or a planned maintenance activity. To know the SLO details for Autonomous AI Database on Dedicated Exadata Infrastructure deployments, see Availability Service Level Objectives (SLOs).

Limits for Autonomous Exadata VM Clusters

You can create multiple Autonomous Exadata VM Clusters (AVMCs) on an Exadata Infrastructure resource. There are no hard limits on the number of AVMCs or Autonomous Container Databases (ACDs) you can provision on your Exadata Infrastructure. The AVMCs and ACDs have a minimum resource requirement, and you can create them as long as the minimum amount of resources is available.

To create an Autonomous Exadata VM Cluster, the minimum resources required are 40 ECPUs per node, 120GB Memory per node, and 338.5GB Local Storage per node, and 6.61TB Exadata Storage. Similarly, the minimum resources needed per node to create an ACD are 8 ECPUs or 2 OCPUs and 50GB Local Storage. As long as your Exadata Infrastructure has these minimum resources available, an AVMC and ACD can be created.

The following example shows the minimum X9M Exadata Infrastructure resources required to provision an AVMC (configured with 2 DB servers) with different number of ACDs.

Note

The default values for Database memory per ECPU (GB) and Database storage (TB) are set to 5GB and 5TB, respectively. However, you can set the Database memory per ECPU to be within a range of 2-5GB.

Property	1 ACD	2 ACDs	3 ACDs	16 ACDs
ECPU Count	80	80	96	512
Memory (GB)	320	320	368	1616
Local Storage (GB)	677	780	883	2222
Exadata Storage (TB)	6.61	6.73	6.86	8.45

Resource Usage Tracking

The compute (CPU) and storage resources allocated to an Autonomous Exadata VM Cluster (AVMC) or Autonomous Container Database (ACD) vary as you provision and run Autonomous AI Databases in them. The number of allocated, provisioned, reserved, reclaimable CPUs and total, available and used storage change through the life-cycle of ACDs and Autonomous AI Databases in an AVMC. As you provision, run, and terminate Autonomous AI Databases or provision, delete, and restart ACDs, the compute and storage resources move into different categories as explained in Compute Management in Autonomous AI Database.

Tracking resource usage for an AVMC or ACD across tenancies is critical in planning the capacity for your Autonomous AI Database on Dedicated Exadata Infrastructure. To simplify resource usage tracking, Oracle Autonomous AI Database provides insights in graphical and tabular formats from the Oracle Cloud Infrastructure (OCI) console.

Autonomous AI Database on Dedicated Exadata Infrastructure supports resource usage tracking at two levels:

- **Autonomous Exadata VM Cluster (AVMC):** You can view the resource usage details of an AVMC from its Details page on the OCI console. In-depth information on the resources is presented in a graphical and tabular format for the AVMC, and each ACD provisioned in the AVMC.
Refer to [View Resource Usage for an Autonomous Exadata VM Cluster](#) for step-by-step instructions and explanation.
- **Autonomous Container Database (ACD):** Similar to AVMC, you can also view the resource usage details of an ACD from its Details page on the OCI console. Comprehensive information on the resources in this ACD is provided on this page in graphical and tabular formats.
Refer to [View Resource Usage for an Autonomous Container Database](#) for step-by-step instructions and explanation.

Resource Usage Visualizations

The resource usage metrics are presented on the OCI console in chart and tabular formats, for Autonomous Exadata VM Cluster (AVMC) and Autonomous Container Database (ACD).

You can access these resource visualizations in either chart or table format the OCI console, by following the instructions outlined in:

- [View Resource Usage for an Autonomous Exadata VM Cluster](#)
- [View Resource Usage for an Autonomous Container Database](#)

✓ Tip

You can choose to see this information either in the graphical or tabular view by selecting **Chart view** or **Table view** from the drop-down list at the top-right corner of this section.

-
- [Autonomous Exadata VM Cluster](#)
 - [Autonomous Container Database](#)

Autonomous Exadata VM Cluster

This section has graphical and tabular representations of the AVMC's resource usage.

Chart View

Chart view is the default view. In this view, you can see 4 graphical visualizations that provide usage details for different resources:



The four charts depict:

1. **CPU usage:** Depicts the Total number of CPUs allocated to this AVMC and how many of those CPUs are reclaimable, available, provisioned, and reserved. This is a doughnut-chart with the total number of CPUs shown in the center of the chart.
2. **CPU usage at VM level:** This is a horizontal bar graph giving a break-down of the CPU usage for each VM in the cluster. You find a horizontal bar for each VM. Each bar shows the number of reclaimable, available, provisioned, and reserved CPUs for that VM with color-coding. Hovering on each colored part of the horizontal bar displays the number of reclaimable, available, provisioned, and reserved CPUs for that specific VM. Clicking the reclaimable, provisioned and reserved bars will open a new panel with the break down of those CPU components by ACDs.
3. **Autonomous Container Database (ACD) usage:** Depicts the Total number of ACDs that can be created in this AVMC along with a break-up of Provisionable ACDs, Provisioned ACDs, and Non-provisionable ACDs. Please see [Resource Terminology](#) to understand what each of these mean. This is a doughnut-chart with the total number of ACDs shown in the center of the chart.
4. **Autonomous AI Database storage (in TBs) usage:** This is a doughnut-chart depicting the available, used, and total Autonomous AI Database storage in TBs. The total storage value is shown in the center of the chart with available and used storage values shown on the chart in different colors.

Table View

To see the resource usage details of an AVMC in table view, select **Table View** from the drop-down list at the top-right corner of the **Resource usage visualizations** section. The table view shows the exact same details as the chart view, in the form of tables.

Resource usage visualizations

Select a view
Table view

Required

CPU usage

Total CPUs	80
Available CPUs	8 CPUs available for allocation to provision new Autonomous AI Databases or scale existing Autonomous AI Databases.
Provisioned CPUs	32 Total CPUs allocated for an Autonomous AI Database within the Autonomous VM Cluster.
Reserved CPUs	40 Total CPUs reserved to support auto scaling of Autonomous AI Databases, Autonomous AI Database failover on node failure, and lifecycle management of empty Autonomous Container Databases.
Reclaimable CPUs	0 Total CPUs from all terminated and scaled down Autonomous AI Databases in all the Autonomous Container Databases within the Autonomous VM Cluster. Reclaimable CPUs are not returned to the Available state until Autonomous Container Database is restarted.

CPU usage at VM level

Click on a bar to see the breakdown of CPU by Autonomous Container Databases

Name ^	Available CPUs	Provisioned CPUs	Reserved CPUs	Reclaimable CPUs	
host-4nosb1	4	18	18	0	...
host-4nosb2	4	14	22	0	...

Page 1 of 1 (1 - 2 of 2 total items) Items per page 10

Autonomous Container Database (ACD) usage

Provisionable	0 Provisionable: Number of Autonomous Container Databases that can be created in the Autonomous VM Cluster.
Provisioned	1 Provisioned: Number of Autonomous Container Databases that have been created in the Autonomous VM Cluster.
Non-Provisionable	2 Non-provisionable: Number of Autonomous Container Databases that cannot be created because of a shortage of available CPUs in the Autonomous VM Cluster.

Autonomous AI Database (ADB) storage (in TBs) usage

Available	8.97 Available: Storage (in TBs) available for the Autonomous AI Database's use.
Used	1.0299997 Used: Storage (in TBs) currently used by the Autonomous AI Database.

The four tables you can see are:

1. **CPU usage:** Lists the number of total, available, provisioned, reserved, and reclaimable CPUs in this AVMC.
2. **CPU usage at VM level:** Lists the number of available, provisioned, reserved, and reclaimable CPUs for each VM in this VM cluster.
3. **Autonomous Container Database (ACD) usage:** Lists the number of provisionable, provisioned, and non-provisionable ACDs in this AVMC.
4. **Autonomous AI Database storage (in TBs) usage:** Shows the available and used Autonomous AI Database storage in TBs.

Autonomous Container Database

This section has graphical and tabular representations of the ACD's resource usage.

Chart View

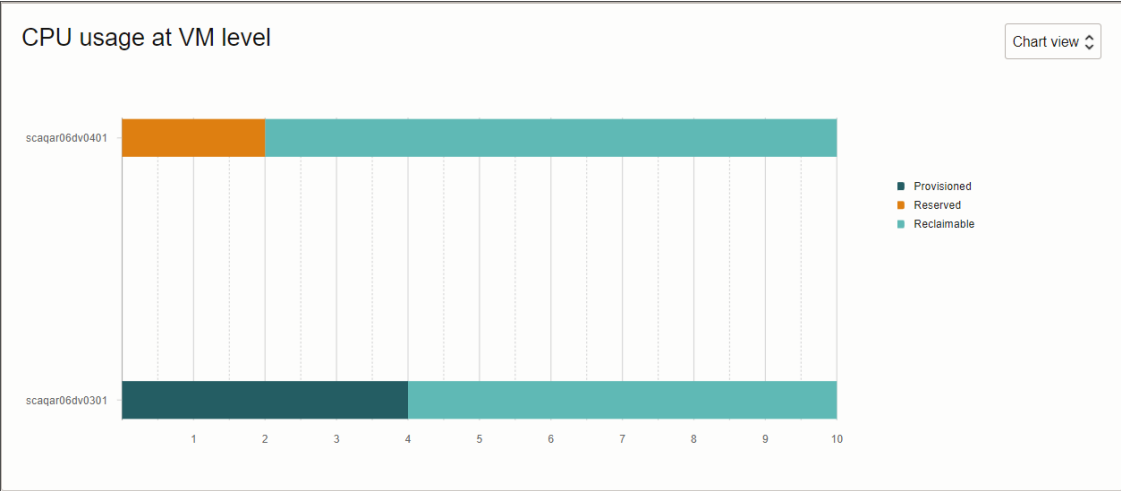


Chart view is the default view. In this view, you can see a horizontal bar graph giving a breakdown of the CPUs used by this ACD in each VM. Each bar shows the number of provisioned, reserved, and reclaimable CPUs for that VM with color-coding. Hovering on each colored part of the horizontal bar displays the number of provisioned, reserved, and reclaimable CPUs for that specific VM.

Table View

CPU usage at VM level				Table view
Name	Provisioned CPUs	Reserved CPUs	Reclaimable CPUs	
scaqar06dv0401	0	2	8	
scaqar06dv0301	4	0	6	
				Showing 2 Items < 1 of 1 >

To see the resource usage details of an ACD in table view, select **Table View** from the drop-down list at the top-right corner of the **CPU usage at VM level** section. The table view shows the exact same details as the chart view, in the form of a table. It lists the number of provisioned, reserved, and reclaimable CPUs for each VM allocated to the ACD from the parent VM cluster.

Exadata System Shapes

Autonomous AI Database on Dedicated Exadata Infrastructure can be provisioned on different Exadata System models such as Oracle Exadata X9M-2, X8M-2, X8-2, or X7-2 system models. Each model comes in different shapes as explained below. Each Exadata System Shape is equipped with a fixed amount of memory, storage, and network resources.

The total resources allocated to your Autonomous AI Database on Dedicated Exadata Infrastructure is determined by the Exadata System (and shape) used to provision your service.

**Tip**

Refer to Characteristics of Infrastructure Shapes to view the specifications of each Exadata System model.

Autonomous AI Database on Dedicated Exadata Infrastructure is offered in the following Exadata System Shapes:

- **Base System:** Contains two database servers and three Oracle Exadata Storage Servers. A Base System is an entry-level configuration. Compared to other configurations, a Base System contains Oracle Exadata Storage Servers with significantly less storage capacity, and database servers with significantly less memory and processing power.
- **Quarter Rack:** Contains two database servers and three Oracle Exadata Storage Servers.
- **Half Rack:** Contains four database servers and six Oracle Exadata Storage Servers.
- **Full Rack:** Contains eight database servers and 12 Oracle Exadata Storage Servers.

X10M systems on Exadata Cloud@Customer deployments are offered in the following Exadata System Shapes:

- **Elastic Base System:** Contains two database servers and three Oracle Exadata Storage Servers and can be expanded up to a maximum of 16 servers by adding additional base database servers and base storage servers. A Base System is an entry-level configuration. Compared to other configurations, a Base System contains Oracle Exadata Storage Servers with significantly less storage capacity, and database servers with significantly less memory and processing power.
- **Elastic:** Starts with two standard memory (1390 GB) database servers and three Oracle Exadata Storage Servers, and can be expanded up to a maximum of 32 database servers and 64 storage servers.
- **Elastic Large:** Starts with two large memory (2090 GB) database servers and three Oracle Exadata Storage Servers, and can be expanded up to 32 large memory database servers and 64 storage servers.
- **Elastic Extra Large:** Starts with two extra large memory (2800 GB) database servers and three Oracle Exadata Storage Servers, and can be expanded up to 32 large memory database servers and 64 storage servers.

Database Observability with Autonomous AI Database Metrics

You can observe the health, capacity, and performance of your Autonomous AI Databases with metrics, alarms, and notifications. You can use Oracle Cloud Infrastructure console or observability APIs to view metrics.

The Autonomous AI Database metrics help you measure useful quantitative data, such as CPU and storage utilization, the number of successful and failed database logon and connection attempts, database operations, SQL queries, and transactions, and so on. You can use metrics data to diagnose and troubleshoot problems with your Autonomous AI Database resources.

Prerequisites

To monitor resources, you must be given the required type of access in a **policy** written by an administrator, whether you're using the Console or the REST API with an SDK, CLI, or another tool.

You need the following policies to access the monitoring services and the resources being monitored:

Allow group `<Group_Name>`
to **read metrics**
in compartment `<Compartment_Name>`

Allow group `<Group_Name>`
to **inspect work-requests**
in `compartment <Compartment_Name>`

✓ Tip

If you try to perform an action and get a message that you don't have permission or are unauthorized, confirm with your administrator the type of access you've been granted and which **compartment** you should work in.

For common policies that give groups access to metrics, see [Let users view metric definitions in a compartment](#) and [Restrict user access to a specific metric namespace](#).

For common policies that give groups access to Notifications, see [Allow a group to manage topics](#), [Allow a group to manage topic subscriptions](#), and [Allow a group to publish messages to topics](#).

Using the Console

You can use Oracle Cloud Infrastructure console to:

- View Autonomous AI Database Metrics for a Database
- View Autonomous AI Database Metrics for Databases in a Compartment

Using the API

You can use the following APIs for monitoring:

- [Monitoring API](#) for metrics and alarms.
- [Notifications API](#) for notifications (used with alarms).

For information about using the API and signing requests, see [REST APIs](#) and [Security Credentials](#). For information about SDKs, see [Software Development Kits and Command Line Interface](#).

Available Metrics: oci_autonomous_database

This topic describes the metrics emitted by the Database service in the oci_autonomous_database namespace.

Database service metrics for Autonomous AI Databases include the following **dimensions**:

- **AUTONOMOUSDBTYPE**
The type of Autonomous AI Database, Autonomous AI Lakehouse or Autonomous AI Transaction Processing.
- **deploymentType**
The Exadata infrastructure type, shared or dedicated. When using the Console to view default metric charts for multiple Autonomous AI Databases, you must specify this dimension.
- **DISPLAYNAME**
The friendly name of the Autonomous AI Database.
- **REGION**
The **region** in which the Autonomous AI Database resides.
- **RESOURCEID**
The **OCID** of the Autonomous AI Database.
- **RESOURCENAME**
The name of the Autonomous AI Database.

The metrics listed in the following table are automatically available for any Autonomous AI Database that you create. You do not need to enable monitoring on the resource to get these metrics.

Note

Valid alarm intervals are 5 minutes or greater due to the frequency at which these metrics are emitted. See [To create an alarm](#) for details on creating alarms.

In the following table, metrics that are marked with an asterisk (*) can be viewed only on the **Service Metrics** page of the Oracle Cloud Infrastructure console. All metrics can be filtered by the dimensions described in this topic.

Metric	Metric Display Name	Unit	Description	Collection Frequency
ApplyLag	Apply Lag	seconds	This metric displays (in seconds) how far the standby database is behind the primary database as of the time sampled. Statistic: Mean Interval: 5 minutes	5 minutes
BlockChanges	DB Block Changes	changes per second	The average number of blocks changed per second. Statistic: Mean Interval: 1 minute	5 minutes

Metric	Metric Display Name	Unit	Description	Collection Frequency
BlockingSessions	Blocking Sessions Count	count	The number of sessions that have been blocking other sessions for more than 60 seconds. Statistic: Max Interval: 1 minute	5 minutes
CpuTime*	CPU Time	seconds per second	Average rate of accumulation of CPU time by foreground sessions in the database over the time interval. Statistic: Mean Interval: 1 minute	5 minutes
CpuUtilization	CPU Utilization	percent	The CPU usage expressed as a percentage, aggregated across all consumer groups. The utilization percentage is reported with respect to the number of CPUs the database is allowed to use. Statistic: Mean Interval: 1 minute	5 minutes
CurrentLogons*	Current Logons	count	The number of successful logons during the selected interval. Statistic: Sum Interval: 1 minute	5 minutes
DBTime*	DB Time	seconds per second	The amount of time database user sessions spend executing database code (CPU Time + WaitTime). DB Time is used to infer database call latency, because DB Time increases in direct proportion to both database call latency (response time) and call volume. It is calculated as the average rate of accumulation of database time (CPU + Wait) by foreground sessions in the database over the time interval. It is also known as Average Active Sessions. Statistic: Mean Interval: 1 minute	5 minutes
ECPUsAllocated	ECPUs Allocated	count	The actual number of ECPUs allocated by the service during the selected interval of time. Statistic: Count Interval: 5 minutes	5 minutes
ExecuteCount	Execute Count	count	The number of user and recursive calls that executed SQL statements during the selected interval. Statistic: Sum Interval: 1 minute	5 minutes

Metric	Metric Display Name	Unit	Description	Collection Frequency
HardParseCount	Parse Count (Hard)	count	Currently allocated CPUs count. Statistic: Sum Interval: 1 minute	5 minutes
IOPS	IOPS	operations per second	The average number of I/O operations per second. Statistic: Mean Interval: 1 minute	5 minutes
IOThroughput	IO Throughput	MB per second	The average throughput in MB per second. Statistic: Mean Interval: 1 minute	5 minutes
LogicalBlocksRead	Logical Reads	reads per second	The average number of logical block reads ("db block gets" plus "consistent gets") per second. Includes buffered and direct I/O. Statistic: Mean Interval: 1 minute	5 minutes
LogicalReads	Session Logical Reads	count	The sum of "db block gets" plus "consistent gets", during the selected time interval. This includes logical reads of database blocks from either the buffer cache or process private memory. Statistic: Sum Interval: 1 minute	5 minutes
OCPUAllocated	OCPUs Allocated	integer	The actual number of OCPUs allocated by the service during the selected interval of time. Statistic: Count Interval: 1 minute	5 minutes
ParsesByType	Parses By Type	parses per second	The number of hard or soft parses per second. Statistic: Mean Interval: 1 minute	5 minutes
ParseCount*	Parse Count (Total)	count	The number of hard and soft parses during the selected interval. Statistic: Sum Interval: 1 minute	5 minutes
ParseFailureCount	Parse Failures Count	count	The number of parse failures during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
PhysicalReads	Physical Reads	count	The number of data blocks read from disk, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes

Metric	Metric Display Name	Unit	Description	Collection Frequency
PhysicalReadTotalBytes	Physical Read Total Bytes	count	The size in bytes of disk reads by all database instance activity including application reads, backup and recovery, and other utilities, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
PhysicalWrites	Physical Writes	count	The number of data blocks written to disk, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
PhysicalWriteTotalBytes	Physical Write Total Bytes	count	The size in bytes of all disk writes for the database instance including application activity, backup and recovery, and other utilities, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
QueuedStatements	Queued Statements	count	The number of queued SQL statements, aggregated across all consumer groups, during the selected interval. Statistic: Sum Interval: 1 minute	1 minute
RedoGenerated	Redo Generated	count	Amount of redo generated in bytes, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
RedoSize	Redo Generated	MB per second	The Average amount of redo generated in MB per second. Statistic: Mean Interval: 1 minute	5 minutes
RunningStatements	Running Statements	count	The number of running SQL statements, aggregated across all consumer groups, during the selected interval. Statistic: Sum Interval: 1 minute	5 minutes
Sessions	Sessions	count	The number of sessions in the database. Statistic: Sum Interval: 1 minute	5 minutes
SessionUtilization	Session Utilization	Percent	The maximum session utilization expressed as a percentage, aggregated across all consumer groups. Statistic: Mean Interval: 1 minute	5 minutes

Metric	Metric Display Name	Unit	Description	Collection Frequency
SQLNetBytesFromClient	SQL Net Bytes Received from Client	count	The number of bytes received from the client over Oracle Net Services, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
SQLNetBytesFromDBLink	SQL Net Bytes Received from DBLink	count	The number of bytes received from a database link over Oracle Net Services, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
SQLNetBytesToClient	SQL Net Bytes Sent to Client	count	The number of bytes sent to the client from the foreground processes, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
SQLNetBytesToDBLink	SQL Net Bytes Sent to DBLink	count	The number of bytes sent over a database link, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
StorageAllocated*	Storage Space Allocated	GB	Maximum amount of space allocated to the database during the interval. Statistic: Max Interval: 1 hour	1 hour
StorageAllocatedByTablespace*	Allocated Storage Space By Tablespace	GB	Maximum amount of space allocated for each tablespace during the interval. Statistic: Max Interval: 1 hour	1 hour
StorageUsed*	Maximum Storage Space	GB	Maximum amount of space used during the interval. Statistic: Max Interval: 1 hour	1 hour
StorageUsedByTablespace*	Storage Space Used By Tablespace	GB	Maximum amount of space used by each tablespace during the interval. Statistic: Max Interval: 1 hour	1 hour
StorageUtilization	Storage Utilization	percent	The percentage of provisioned storage capacity currently in use. Represents the total allocated space for all tablespaces. Statistic: Mean Interval: 1 hour	1 hour

Metric	Metric Display Name	Unit	Description	Collection Frequency
StorageUtilizationByTablespace*	Storage Space Utilization By Tablespace	percent	The percentage of space utilized by each tablespace. Statistic: Mean Interval: 1 hour	1 hour
TransactionsByStatus	Transactions By Status	transactions per second	The number of committed or rolled back transactions per second. Statistic: Mean Interval: 1 minute	5 minutes
TransactionCount*	Transaction Count	count	The combined number of user commits and user rollbacks during the selected interval. Statistic: Sum Interval: 1 minute	5 minutes
TransportLag	Transport Lag	seconds	The approximate number of seconds of redo not yet available on the standby database as of the time sampled. Statistic: Mean Interval: 5 minutes	5 minutes
UserCalls*	User Calls	count	The combined number of logons, parses, and execute calls during the selected interval. Statistic: Sum Interval: 1 minute	5 minutes
UserCommits	User Commits	count	The number of user commits during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
UserRollbacks	User Rollbacks	count	Number of times users manually issue the ROLLBACK statement or an error occurs during a user's transactions, during the selected time interval. Statistic: Sum Interval: 1 minute	5 minutes
WaitTime*	Wait Time	seconds per second	Average rate of accumulation of non-idle wait time by foreground sessions in the database over the time interval. The wait time component of Average Active Sessions. Statistic: Mean Interval: 1 minute	5 minutes

Database Observability with Performance Hub

You can use Performance Hub to observe your databases for defined time periods and download statistical reports. Performance Hub lets you view real-time and historical performance data for an Autonomous AI Database.

✓ **Tip**

To try out Performance Hub, you can go through *Lab 10: Managing Database Performance* in the Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop.

You can access Performance Hub from the Autonomous AI Database **Details** page. To access Performance Hub:

1. Go to the **Details** page of the Autonomous AI Database you want to monitor with Performance Hub.

For instructions, see [View Details of a Dedicated Autonomous AI Database](#).

2. On the **Details** page, click **Performance Hub**.

Performance Hub provides the following options to monitor database activity, diagnose issues, and tune queries to improve the performance of your database.

Name	Description
Quick Select	Used to quickly set the time range to Last Hour, Last 8 Hours, Last 24 Hours, or Last Week.
Time Range	Use the date selector to set the time range for database observability with Performance Hub.
Time Zone	Select an entry from this list to to have all times based on UTC (Coordinated Universal Time) time, your local web browser time, or the time zone setting of the database.
Activity Summary	The Activity Summary (Average Active Sessions) timeline shows active sessions during the selected time range. It displays the average number of active sessions broken down by CPU, User I/O, and Wait. It also shows the Max CPU usage. You can hide or show the activity summary timeline by selecting or deselecting Hide Activity Summary located next to Time Zone above this timeline.

Name	Description
Reports	The Reports drop down menu allows for generation of the following types of performance diagnostic reports. Automatic Workload Repository (AWR) report: The AWR report collects, processes, and maintains performance statistics for problem detection and self-tuning purposes. This data is both in memory and stored in the database. An AWR report shows data captured between two points in time (or snapshots). AWR reports are divided into multiple sections. The content of the report contains the workload profile of the system for the selected range of snapshots. The HTML report includes links that you can use to navigate quickly between sections. Active Session History (ASH) report: The ASH report contains data that helps you triage transient performance problems and perform scoped or targeted performance analysis by various dimensions such as time, session, module, action, or SQL identifier. This report type requires enablement of Database Management. See Enable Diagnostics & Management for Autonomous AI Databases for instructions.
Active Session History (ASH) Analytics	The ASH Analytics tab is displayed by default. It shows Active Session History (ASH) analytics charts to explore the Active Session History data. You can drill down into database performance across multiple dimensions such as Consumer Group, Wait Class, SQL ID, and User Name. Select an Average Active Sessions dimension and view the top activity for that dimension for the selected time period. For more information, see ASH Analytics.
SQL Monitoring	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. For more information, see SQL Monitoring.

Name	Description
Automatic Database Diagnostic Monitor (ADDM)	The ADDM tab provides access to analysis information gathered by the Automatic Database Diagnostic Monitor (ADDM) tool. ADDM analyzes AWR (Automatic Workload Repository) snapshots on a regular basis, locates root causes of any performance problems, provides recommendations for correcting the problems, and identifies non-problem areas of the system. AWR is a repository of historical performance data, therefore ADDM can analyze performance issues after the event, often saving time and resources in reproducing a problem. For more information, see Automatic Database Diagnostic Monitor (ADDM).
Workload	Use the Workload tab to visually monitor the database workload to identify spikes and bottlenecks. This tab shows four chart areas that show database workload in various ways: • CPU Statistics: Charts CPU usage. • Wait Time Statistics: Shows the wait time across the database's foreground sessions, divided by wait classes. • Workload Profile: Charts user (client) workload on the database. • Sessions: Shows the number of sessions, successful logons, and current logons. For more information, see Workload.
Blocking Sessions	This tab hierarchically lists sessions that are waiting or are blocked by sessions that are waiting. You can set the minimum wait time required for sessions to be displayed in the list, and you can view a variety of information about a session to determine whether to let it continue or to kill it. For more information, see Blocking Sessions.

For detailed guidance on viewing and monitoring your database performance using Performance Hub, refer to [Using Performance Hub to Analyze Database Performance](#).

Database Observability with Dynamic Performance Views

Autonomous AI Database on Dedicated Exadata Infrastructure exposes a set of dynamic performance views. These views allow database administrators to monitor the real-time and historical performance of their databases comprehensively, and are also available at the Autonomous Container Database (ACD) level.

Dynamic Performance Views

Dynamic performance views offer deep insights into the performance and health of Autonomous AI Databases, enabling database administrators to perform detailed analyses and optimizations. ACD_ performance views allow you to monitor databases at the Autonomous Container Database level. You can leverage real-time and historical data for effective database monitoring and management provided that you have the privileges to access these views.

Dynamic performance views are continuously updated system views maintained by the database server. They provide real-time data primarily related to database performance, making them invaluable for database monitoring and tuning.

Using dynamic performance views, you can:

- Monitor real-time database activities through various wait events and classes.
- Analyze historical performance data to identify trends or issues.
- Assess database resource utilization against limits.
- Summarize performance metrics to gauge overall database health.

Accessing Dynamic Performance Views at the Autonomous Container Database Level

In Autonomous AI Database instances, besides accessing standard PDB level Oracle Database dynamic performance views, you can also access views at the Autonomous Container Database level which are prefixed with ACD_, followed by either V\$ for single-instance databases or GV\$ for Real Application Clusters (RAC), indicating global views that aggregate data across all instances.

ADMIN privileges are necessary to query the Dynamic performance views. You can use GRANT command to assign READ privilege on the Dynamic performance views for non-administrative users.

```
GRANT READ ON ACD_V$SYSMETRIC TO non_admin_user;
```

Supported Dynamic Performance Views at the Autonomous Container Database Level

The following table lists the Autonomous Container Database level dynamic performance views available in Autonomous AI Database on Dedicated Exadata Infrastructure:

View Name	Description	Snapshot
ACD_V\$EVENTMETRIC ACD_GV\$EVENTMETRIC	Real-time wait event metrics	V\$EVENTMETRIC
ACD_V\$RESOURCE_LIMIT ACD_GV\$RESOURCE_LIMIT	Current resource usage and limits	V\$RESOURCE_LIMIT
ACD_V\$SYSMETRIC ACD_GV\$SYSMETRIC	Real-time system performance metrics	V\$SYSMETRIC
ACD_V\$SYSMETRIC_HISTORY ACD_GV\$SYSMETRIC_HISTORY	Historical system performance metrics	V\$SYSMETRIC_HISTORY
ACD_V\$SYSMETRIC_SUMMARY ACD_GV\$SYSMETRIC_SUMMARY	Summary of system performance metrics	V\$SYSMETRIC_SUMMARY
ACD_V\$WAITCLASSMETRIC ACD_GV\$WAITCLASSMETRIC	Metrics by wait class	V\$WAITCLASSMETRIC

View Name	Description	Snapshot
ACD_V\$WAITCLASSMETRIC_HISTORY ACD_GV\$WAITCLASSMETRIC_HISTORY	Historical wait class metrics	V\$WAITCLASSMETRIC_HISTORY
ACD_DBA_HIST_SYSMETRIC_SUMMARY	History of statistical summary of system performance metrics	DBA_HIST_SYSMETRIC_SUMMARY

The following Oracle Exadata Database Machine views are supported on Autonomous AI Database on Dedicated Exadata Infrastructure. Each of these views begins with the prefix V\$CELL and contain internal statistical information. For additional information about these views, see [About the V\\$CELL_VIEWS](#).

View	Description
ACD_V\$CELL_GLOBAL ACD_GV\$CELL_GLOBAL	Global statistics for Exadata cell
ACD_V\$CELL_IOREASON ACD_GV\$CELL_IOREASON	I/O reasons and metrics for Exadata cell
ACD_V\$CELL_IOREASON_NAME ACD_GV\$CELL_IOREASON_NAME	Names and descriptions of I/O reasons for Exadata cell
ACD_V\$CELL_METRIC_DESC ACD_GV\$CELL_METRIC_DESC	Descriptions of metrics for Exadata cell
ACD_V\$CELL_GLOBAL_HISTORY ACD_GV\$CELL_GLOBAL_HISTORY	Historical information for Exadata cell

For a complete list and detailed information about each dynamic performance view, refer to the Oracle documentation on [About Dynamic Performance Views](#).

Example Queries

To help you get started, see the example queries to extract metrics from these views:

Query System Metrics

```
SELECT METRIC_NAME, VALUE, METRIC_UNIT
FROM ACD_V$SYSMETRIC
WHERE METRIC_NAME IN ('Database CPU Time Ratio', 'Database Wait Time Ratio');
```

Query Resource Limits

```
SELECT RESOURCE_NAME, CURRENT_UTILIZATION, MAX_UTILIZATION, LIMIT_VALUE
FROM ACD_V$RESOURCE_LIMIT
WHERE RESOURCE_NAME IN ('processes', 'sessions', 'enqueue_locks');
```

Query Wait Events

```
SELECT e.NAME AS EVENT_NAME, m.NUM_SESS_WAITING, m.TIME_WAITED
FROM ACD_V$EVENTMETRIC m JOIN V$EVENT_NAME e ON m.EVENT_ID = e.EVENT_ID
```



```
WHERE e.NAME IN ('db file sequential read', 'db file scattered read', 'log  
file sync');
```

Query Exadata Cell Global Metrics

```
SELECT METRIC_NAME, METRIC_VALUE, METRIC_TYPE  
FROM ACD_V$CELL_GLOBAL  
WHERE METRIC_NAME IN (  
    'Cumulative User CPU ticks',  
    'Cumulative System CPU ticks',  
    'Network bytes received',  
    'Network bytes transmitted',  
    'Total Reads',  
    'Total Writes',  
    'Ticks spent reading',  
    'Ticks spent writing');
```

Query I/O Reasons in Exadata Cells

```
SELECT REASON_NAME, METRIC_NAME, METRIC_VALUE, METRIC_TYPE  
FROM ACD_V$CELL_IOREASON  
WHERE CELL_NAME = 'YourCellName';
```

Use Database Actions to Manage and Observe Databases

✓ Tip

For a "try it out" alternative that demonstrates these instructions, run *Lab 7: Database Actions Console* in the Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop.

Oracle Database Actions, a browser-based application that provides many database development, management and monitoring features, is built into Autonomous AI Database on Dedicated Exadata Infrastructure. Therefore, you can use it without downloading or installing additional software on your system. Database Actions was earlier known as SQL Developer Web.

① Note

Database Actions can not be launched from a standby database in an Autonomous Data Guard configuration.

Use Database Actions monitoring features, like Performance Hub, to view and analyze performance of your database. To connect to Database Actions, see [Connect with Oracle Database Actions](#).

Alert Logs are available in Database Actions. You can query `v$diag_alert_ext` to get the log data; for example, if you want to export the data from these logs to another analytic solution for tracing ORA errors such as ORA-600, ORA-7445, and so on. However, please note that this information is provided only for your awareness, as Oracle will respond to these issues and provide a resolution.

For information about the wide-ranging features of Database Actions, see About Database Actions.

Use Database Management Service to Observe Databases

APPLIES TO: • Oracle Public Cloud only

You can use Database Management Service to observe the health of a single Autonomous AI Database or a fleet of Autonomous AI Databases.

The Diagnostics & Management features available in the Database Management service assist with advanced monitoring and management of Autonomous AI Databases. Using a private endpoint, you can configure network access between Database Management and an Autonomous AI Database and use features such as AWR Explorer, SQL Tuning Advisor, and Performance Hub - Top Activity Lite.

Before you enable Database Management Diagnostics & Management for Autonomous AI Databases, you must:

- Complete the general Oracle Cloud Infrastructure prerequisite tasks listed in [Perform General Oracle Cloud Infrastructure Prerequisite Tasks](#).
- Obtain the permissions required to enable Diagnostics & Management for Autonomous AI Databases, as detailed in [Permissions Required to Enable Diagnostics & Management for Autonomous AI Databases](#).
- Complete the Autonomous AI Database-specific prerequisite tasks listed in [Autonomous AI Database-related Prerequisite Tasks](#).

Note

Certain metrics and Performance Hub features continue to be available free of cost for Autonomous AI Databases. However, you must enable Database Management to use the advanced monitoring and management features, which are available at an additional cost.

For information on how to enable Database Management Diagnostics & Management for Autonomous AI Databases, see [Enable Diagnostics & Management for Autonomous AI Databases](#).

After enabling the Diagnostics & Management features in Database Management, you can use the full suite of available capabilities to monitor and manage a single Autonomous AI Database or a fleet of Autonomous AI Databases. Here are some of the important performance monitoring and administration-related features:

- Database fleet overview to monitor multiple databases in one location
- Performance Hub for real-time monitoring of the database workload
- AWR Explorer to visualize AWR data over time
- SQL Monitoring to view top and current running SQL statements with detailed execution statistics
- SQL performance tuning to optimize database performance
- AWR and ADDM reports for in-depth performance analysis
- User administration to track configuration and details of database users

- Parameter configuration to update and monitor database configuration parameters
- Database jobs to streamline collections and run at scale utilizing Database Groups

For more information on these features and how to use them, see:

- [Monitor the Health of Your Oracle Database Fleet](#)
- [Monitor and Manage a Specific Managed Database](#)
- [Diagnostics & Management Feature Support Matrix for Oracle Databases](#)

① Note

If you enabled Diagnostics & Management for Autonomous AI Databases on the **Autonomous AI Database details** page or in Database Management prior to September 2024, the advanced monitoring and management features may not be available. To enable and use these features for Autonomous AI Databases, you must provide the connection information in the **Edit Database Management** panel. For information, see [Enable Diagnostics & Management for Autonomous AI Databases](#).

Use Enterprise Manager to Manage and Observe Databases

You can use Oracle Enterprise Manager to manage and observe Autonomous AI 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.

① Note

Enterprise Manager version 13.3 and above support monitoring and management of Autonomous AI Databases.

To use Enterprise Manager to manage and monitor your 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 AI Database.
2. Use the Enterprise Manager console, CLI or REST API to discover the database and add it as a target.

For detailed steps to perform these tasks, see [Discover Autonomous AI Databases](#) in *Enterprise Manager Cloud Control Administrator's Guide for Oracle Autonomous AI Databases*.

 Tip

For a "try it out" alternative that demonstrates how to connect to an Autonomous AI Database from Oracle Enterprise Manager, see *Lab 12: Deploy OEM and Connect ADB to OEM* in Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop.

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 AI Databases*.

Use Ops Insights on Autonomous AI Database

APPLIES TO:  Oracle Public Cloud only

Ops Insights is a cloud-native service that enables users to make informed, data-driven, Autonomous AI Database resource and performance management decisions.

About Ops Insights

Ops Insights provides 360-degree insight into the resource utilization and capacity of databases and hosts. You can easily analyze CPU and storage resources, forecast capacity issues, and proactively identify SQL performance issues across your database fleet.

With Ops Insights, you can:

- Analyze resource usage of databases/hosts across the enterprise.
- Forecast future demand for resources based on historical trends.
- Compare SQL Performance across databases and identify common patterns.
- Identify SQL performance trends across enterprise-wide databases.
- Analyze AWR statistics for database performance, diagnostics, and tuning across a fleet of databases.

To use Ops Insights with Autonomous AI Database, you must first enable it from the Oracle Cloud Infrastructure console. See [Enable Ops Insights for a Dedicated Autonomous AI Database](#).

Ops Insights consists of the following integrated applications:

- Capacity Planning
- Oracle SQL Warehouse
- Exadata Insights
- AWR Hub

These applications allow application DBAs, DevOps, and IT executives to make critical decisions about their databases and hosts using historical and long term data. To explore

these features, see [Get Started with Ops Insights](#) in *Oracle Cloud Infrastructure Documentation*.

Use Work Requests to Observe Long-Running Operations

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure 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 AI 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 AI Database on Dedicated Exadata Infrastructure 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](#).

Use Events and Notifications to Watch Resource Activity

When Oracle Autonomous AI Database on Dedicated Exadata Infrastructure performs operations on 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.

✓ Tip

See Notifications Use Case: Emails for Maintenance Events for a short end-to-end example of using the Events and Notifications services in this way.

✓ Tip

For a "try it out" demonstration for setting up Autonomous AI Database event notifications through OCI Notification Service, see *Lab 11: OCI Notification Service* in Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop.

This combination of Autonomous AI Database events, the Events service and the Notification service provide a very rich set of options for keeping abreast of resource activity because:

- Autonomous AI Database generates events for more than forty different operations, providing you fine-grained choices for resource tracking.

For listings of the operations that generate events for Autonomous AI Database on Oracle Public Cloud and Multicloud, see *Events for Autonomous AI Database on Dedicated Exadata Infrastructure*.

For listings of the operations that generate events for Autonomous AI Database on Exadata Cloud@Customer, see *Oracle Exadata Cloud@Customer Events*.

- The Events service supports arbitrary grouping of events in a single rule, providing flexible aggregation of the events you are interested in.

For more information about how the Events service works and how to set up the rules and actions it uses, see *Overview of Events*.

- The Notifications service supports sending notices to email, PagerDuty, Slack, custom HTTPS urls or Oracle Functions, providing you a wide variety of communication channels.

For more information about how the Notifications service works and how to set up the topics and subscriptions it uses, see [Notifications Overview](#).

Note

All the normal service events, metrics and logs that apply to Oracle Public Cloud also apply to Multicloud deployments. Additionally, for AWS Multicloud deployments, you can use the monitoring and management services in Oracle Database@AWS. See [Monitoring services in Oracle Database@AWS](#) for details.

Observe Autonomous AI Database on Dedicated Exadata Infrastructure Resource Logs with OCI Logging Service

The Oracle Cloud Infrastructure Logging service is a highly scalable and fully managed single pane of glass for all the logs in your tenancy.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure is an Oracle Cloud Infrastructure service that emits service logs, and you can enable or disable capturing these logs using the OCI Logging Service.

Logging provides access to logs from Oracle Cloud Infrastructure resources. These logs include critical diagnostic information that describes how resources are performing and being accessed. Refer to the [OCI Logging Service](#) for more details.

Privileges needed to view logs for Autonomous AI Database resources:

- For an Autonomous Container Database (ACD), the user must have the `AUTONOMOUS_CONTAINER_DATABASE_INSPECT` privilege.
- For an Autonomous AI Database, the user must have the `AUTONOMOUS_DATABASE_CONTENT_READ` privilege.
- For an Autonomous VM Cluster on Oracle Public Cloud deployments, the user must have the `CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT` privilege.
- For an Autonomous VM Cluster on Exadata Cloud@Customer deployments, the user must have the `AUTONOMOUS_VM_CLUSTER_INSPECT` privilege.

List of Autonomous AI Database on Dedicated Exadata Infrastructure Logs

Table 3-5 Autonomous AI Database Logs

Log	Scope	Description
Unified audit Logs	Autonomous Container Database (ACD)	In Autonomous AI Database, unified audit records are periodically collected from the UNIFIED_AUDIT_TRAIL view at ten-minute intervals and written to an audit log file. This log file is then transmitted by the log transmitter service to an Oracle Cloud Infrastructure (OCI) Object Storage bucket. The audit logs are subsequently ingested into OCI Logging, making the audit data available for centralized monitoring, search, and analysis. Unified audit logs are JSON formatted files and can be extracted from the following path <code>/var/opt/oracle/log/unified_audit_logs/</code> You can view the sample log files for Unified audit logs.

Table 3-5 (Cont.) Autonomous AI Database Logs

Log	Scope	Description
Attention Logs	Autonomous Container Database (ACD)	i Not e Attention Logs are available on Oracle Database version 26ai or later. The attention log is a JSON formatted file to capture critical events, making system diagnostics easier. There is a single attention log for each ACD. The attention log is extracted from the following path : <code>\$ORACLE_BASE/diag/rdbms/<ORACLE_UNQNAME>/\$ORACLE_SID/log/attention.log</code>
Alert Logs	Autonomous Container Database (ACD)	Alert logs contain important information about error messages and exceptions that occur during database operations. There is a single alert log for each ACD. The alert log is extracted from the following path: <code>\$ORACLE_BASE/diag/rdbms/<ORACLE_UNQNAME>/\$ORACLE_SID/alert_\$ORACLE_SID.log</code>

Table 3-5 (Cont.) Autonomous AI Database Logs

Log	Scope	Description
Migration Logs	Autonomous AI Database	The migration logs are the logs for Oracle Data Pump export and import operations. The data pump export and import logs are extracted from the following path: <code>/u02/data/dbfs/\$CDB_NAME/\$PDBGUID/import*.log</code> <code>/u02/data/dbfs/\$CDB_NAME/\$PDBGUID/export*.log</code>
ORDS Connection Logs	Autonomous VM Cluster (AVMC)	The ORDS connection log is for monitoring operations related to connecting your Autonomous AI Database to Oracle REST Data Services. The ORDS connection log is extracted from the following path: <code>/u02/app/oracle/product/ords/log/access/</code>
Mongo Connection	Autonomous VM Cluster (AVMC)	The Mongo connection log is for monitoring operations related to connecting to your Autonomous AI Database via Oracle Database API for MongoDB. The Mongo connection log is extracted from the following path: <code>/u02/app/oracle/product/ords/log/mongo/</code>

Table 3-5 (Cont.) Autonomous AI Database Logs

Log	Scope	Description
Network Logs	Autonomous VM Cluster (AVMC)	Network logs include Listener Logs and Scan Listener logs to monitor network activity. Network logs are extracted from the following paths: \$ORACLE_BASE/diag/ tnslsnr/\$HOSTNAME/ listener/trace/ listener.log \$ORACLE_BASE/diag/ tnslsnr/\$HOSTNAME/ listener_scan2/trace/ listener_scan2.log \$ORACLE_BASE/diag/ tnslsnr/\$HOSTNAME/ listener_scan3/trace/ listener_scan3.log

Note

Attention Logs and Alert logs are collectively known as Database Logs.

You can view the sample log files for [Database Logs and Migration Logs](#).

For more information on using Oracle Data Pump to export data from Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, See [Export Data Using Oracle Data Pump](#). For more information on the logs generated at the Autonomous Exadata VM Cluster and Exadata Infrastructure levels, See [Auditing Capabilities in Autonomous AI Database](#).

Prerequisites to Enable OCI Logs

To enable service logs in a resource, a user must be granted manage access in the log group and access to the resource. In general, inspect access for the resource is enough, but check for specific resources. Inspect access provides permission to update the resource and permission for the log group that contains the log. In addition to the permissions to work with the log group, you must have the update permission for the resource, to add service logs to a resource. See [Required IAM Policy](#) for more details.

Enable Logging for an Autonomous AI Database Resource

You can enable logging for an AVMC, ACD, or an Autonomous AI Database. The procedure to enable logs for an ACD or Autonomous AI Database is the same as the procedure followed for any OCI resource. See [Enabling Logging for a Resource](#) for more details.

- From the **Logs** section in the Logging service, click **Enable Service Log**.

- Choose your compartment, select **Autonomous AI Database Dedicated** as the service, and choose the corresponding resource (the specific ACD or Autonomous AI Database).
- For ACD the **Log Category** must be *Database Logs* and for an Autonomous AI Database it must be *Migration Logs*.
- Provide a name for your log file.

For an AVMC resource, you can create a Connection log or a Network log, which includes Listener and Scan logs. For an ACD resource on database versions 26ai onwards, you can create a database log which consists of an Attention log and an Alert log. For an Autonomous AI Database, you can create a migration log which consists of data pump logs.

 Note

Attention logs are available from database versions 26ai onwards.

Viewing Logs for an Autonomous AI Database Resource

You can search and filter the Database logs and Migration logs of your Autonomous AI Database resource from the OCI console.

- Open the navigation menu and select **Observability & Management**. Under **Logging**, select **Search**.
- You can use custom filters. Additionally, you can search for logs based on their type. You can also filter and limit logs based on time.
- The log data in the **Explore** and **Visualize** tabs is reloaded according to your filter settings.

See [Search Logs using the OCI console](#) for more details.

Security

Security Features in Autonomous AI Database on Dedicated Exadata Infrastructure

This article describes the key security features in Autonomous AI Database on Dedicated Exadata Infrastructure.

Note that throughout this section the term "you" is broadly used to mean any administrator in your organization who has the responsibility for performing certain tasks. In some cases, that's the fleet administrator, in others it's the database administrator.

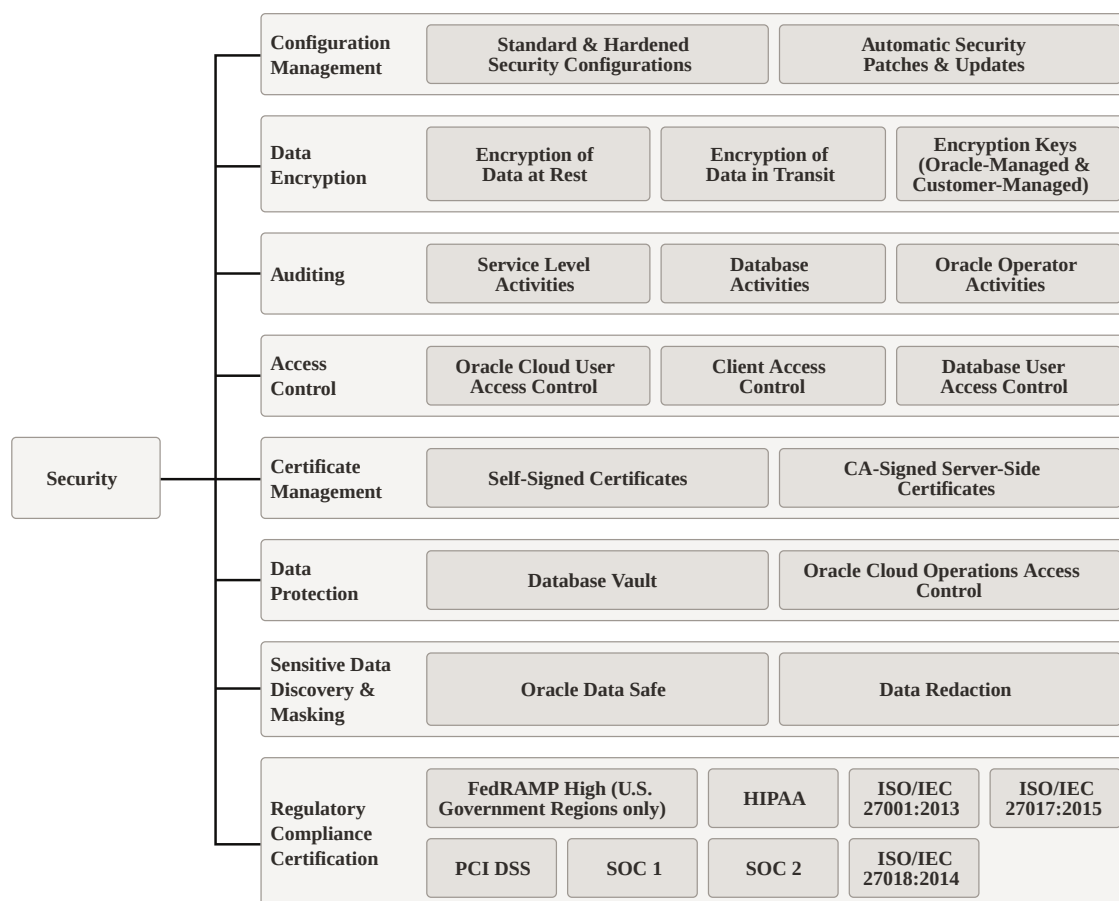
Along with the standard security features of the Oracle AI Database such as privilege analysis, network encryption, centrally managed users, secure application roles, transparent sensitive data protection, and others, a Dedicated Autonomous AI Database adds Database Vault, Data Safe, and other advanced security features at no additional cost.

You can see the building blocks for the key security features of a Dedicated Autonomous AI Database depicted below.

 Tip

In the following image, you can click the feature you want to explore further.

Figure 3-1 Key Security Features



Configuration Management

Built on Oracle Cloud Infrastructure, Autonomous AI Database provides standard, hardened security configurations so you and your team don't need to spend huge amounts of time and money managing configurations across your Autonomous AI Database fleet. All service accounts such as SYS and System are rotated every 90 days. Refer to Configuration Management in Autonomous AI Database to explore further.

Security patches and updates are done automatically, so you don't need to worry about keeping security up to date. These capabilities protect your highly sensitive databases and data from costly and potentially disastrous security vulnerabilities and breaches. Refer to Service Maintenance of Dedicated Autonomous AI Database for more details.

Data Encryption

Autonomous AI Database 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.

Autonomous AI Database uses always-on encryption that protects data at rest and in transit. All data stored in and network communication with Oracle Cloud is encrypted by default. Encryption cannot be turned off.

Refer to Data Encryption in Dedicated Autonomous AI Database for more details about data encryption and master encryption keys.

Auditing

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

Refer to Auditing Capabilities in Dedicated Autonomous AI Database for more details.

Access Control Within Autonomous AI Database on Dedicated Exadata Infrastructure

When configuring Autonomous AI Database on Dedicated Exadata Infrastructure, you need to ensure that your cloud users have access to use and create only the appropriate kinds of cloud resources to perform their job duties. Additionally, you need to ensure that only authorized personnel and applications have access to the Autonomous AI Databases created on dedicated infrastructure. Otherwise, you run the risk of "runaway" consumption of your dedicated infrastructure resources or inappropriate access to mission-critical data.

Before you begin creating and using the cloud resources that provide the dedicated infrastructure feature, you need to formulate an access control plan, and then institute by creating appropriate IAM (Identity and Access Management) and Networking resources. Accordingly, access control within an Autonomous AI Database is implemented at various levels:

- Oracle cloud user access control
- Client access control
- Database user access control

Certificate Management

When a client attempts to connect to an Autonomous AI Database through a TCPS (Secure TCP) database connection service, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure uses standard TLS 1.2 and TLS 1.3 certificate-based authentication to authenticate the connection. However, TLS 1.3 is only supported on Oracle AI Database 23ai or later. Regardless of whether the client attempts to connect through a TCPS or TCP database connection service, the client's access to the database is restricted by the access rights of the database user the client uses to connect.

Oracle Managed Self-signed Certificate

By default, Autonomous AI Database uses *self-signed certificates*. A self-signed certificate is a security certificate that is system generated.

Certificate generation

Oracle managed self-signed certificates are automatically generated while provisioning an Autonomous Exadata VM Cluster (AVMC) and apply to all the databases created in that AVMC.

Certificate management

Self-signed certificates are generated and associated with an AVMC automatically. However, you must download Autonomous AI Database client wallet before connecting to your databases. With self-signed certificates, connecting to databases without a wallet is not an option. Refer to Download Client Credentials for instructions on downloading a wallet for your database.

Depending on the requirement, either of the following certificate types are associated with your AVMC:

- **Database SSL certificate:** SSL certificate for database client connections.
- **ORDS SSL certificate:** SSL certificate for Application Express (APEX) applications.

Certificate rotation

To meet your organization's security compliance needs, you can rotate the Oracle-managed self-signed certificates using the Oracle Cloud Infrastructure (OCI) console or API. Refer to Manage Security Certificates for an Autonomous Exadata VM Cluster Resource for step-by-step instructions. This is called certificate rotation.

For newly provisioned Autonomous Exadata VM Cluster (AVMC) resources, Oracle-managed self-signed certificates have a 13-month validity from their creation. Rotating an SSL certificate using the console or API rotates both the server-side and client-side certificates and resets their validity to 13 months. When an Oracle-managed server-side or client-side certificate is not rotated before its expiry, Oracle rotates them automatically and generates new wallets.

In the case of **database SSL certificates**, certificate rotation does not invalidate the existing certificate immediately.

Within two weeks from the certificate rotation, you can connect to your databases using the Autonomous AI Database client wallet you downloaded before or after the certificate rotation.

After two weeks from the certificate rotation:

- The database wallet *downloaded before the certificate rotation* is invalidated and can not be used to connect to your databases.
- The database wallet *downloaded within two weeks* from the certificate rotation remains active and can be used to connect to your databases.
- Any new database wallet *downloaded after two weeks* from the certificate rotation can be used to connect to your databases.

Let's discuss this with an example:

Consider an SSL certificate, say C1, is due to expire and you have rotated this certificate on the 1st of February. For two weeks from Feb 1st, that until Feb 14th, the old certificate (C1) is still available to use. You can either continue using the old certificate (C1) or download a new database wallet for the rotated certificate (C2) for your database connections. After two weeks from Feb 1st, that is, from Feb 14th onwards, the old certificate (C1) gets invalidated and can not be used for database connections. You can continue using the database wallet that you downloaded post certificate rotation (C2) during these two weeks. You can also download a new database wallet and start using it for your database connections after two weeks from rotation.

You can also rotate a database SSL certificate *within two weeks* from its most recent rotation. In this scenario, the old certificate (about to get invalidated due to the first rotation) gets disabled immediately. The next certificate (resulting from the first rotation) remains active, and a third certificate (resulting from the second rotation) will be waiting for activation for *two weeks from the second rotation*. Any database wallet downloaded before the first rotation gets invalidated soon after the second rotation. You can continue connecting to the database with any database wallet downloaded after the first rotation until two weeks from the second rotation. After the completion of two weeks from the second rotation, you can only connect to your databases using a client wallet downloaded after the second rotation, that is, a wallet that was downloaded within two weeks from the second rotation or later.

In the above example, if you rotate the same certificate (C1) again *within two weeks* from Feb 1st, the certificate undergoes double rotation. In this case, the old certificate (certificate before the first rotation, that is, C1) gets invalidated immediately. The certificate resulting from the first

rotation (C2) remains active and a third certificate resulting from the second rotation, say C3, will be waiting for activation for *two weeks from the second rotation*. After two weeks from the second rotation, the certificate resulting from the first rotation (C2) also gets invalidated and any database wallets downloaded before the second rotation can not be used for database connections. You can continue using the database wallet that you downloaded post certificate rotation (C3) during these two weeks. You can also download a new database wallet and start using it for your database connections after two weeks from the second rotation.

In the case of **ORDS SSL certificates**, all the existing application connections would be lost with the certificate rotation and you are advised to restart ORDS. The two week buffer period discussed above does not apply when you rotate an ORDS SSL certificate.

Bring Your Own Certificate (BYOC)

Bring Your Own Certificate (BYOC) lets you use your *CA-signed server-side certificate* with your Autonomous AI Databases.

Certificate generation

To bring your own certificate, you must first create the certificate using the Oracle Cloud Infrastructure (OCI) Certificate Service as demonstrated in [Creating a Certificate](#). These certificates must be signed and must be in the PEM format, that is, their file extension must be .pem, .cer, or .crt only.

Certificate installation

After creating the CA-signed server-side certificate, you must install it with your AVMC so that any databases created in it can use this certificate for secure connections. Associating your BYOC certificate with an AVMC is facilitated from the **Manage Certificates** dialog on the OCI console. On this dialog, choose the **Bring your own certificate**, and select the certificate that you would have created earlier from the select list. Optionally, you can also specify a **CA certificate** with certificate authority and CA bundle. However, if you are selecting a CA bundle for an ORDS SSL certificate, the bundle must only contain certificates which are part of the certificate chain. Refer to Manage Security Certificates for an Autonomous Exadata VM Cluster Resource for step-by-step instructions.

Certificate management

You can connect to Autonomous AI Databases associated with a CA-signed server-side with or without an Autonomous AI Database client wallet.

- To connect to your database using a database wallet, you must first download the client credentials from the database **Details** page using the OCI console. Refer to Download Client Credentials for instructions.
- To connect to your database without a client wallet, you must ensure that:
 - One-way TLS connections are enabled at the AVMC level. This is a setting defined using the **Listener** parameter in **Advanced options** while provisioning the AVMC. Refer to Create an Autonomous Exadata VM Cluster for guidance.
 - The CA-signed server-side certificate is signed by a well known public CA so that it is trusted by the client operation system by default.

Certificate rotation

To meet your organization's security compliance needs, you can rotate the CA-signed server-side certificates using the Oracle Cloud Infrastructure (OCI) console or API. Refer to Manage Security Certificates for an Autonomous Exadata VM Cluster Resource for step-by-step instructions.

You must rotate them before their expiry date, failing which the databases on this AVMC will not be reachable on TLS ports until you provide a valid certificate. However, you can continue to access the databases on a non-TLS port such as 1521.

Certificate Events

The following events are published for security certificate management:

Event	Generated when
sslcertificateexpiry.reminder	An Autonomous Exadata VM Cluster determines that a wallet is due to expire in less than six (6) weeks. This event is reported at most once per week. This event is triggered when there is a connection that uses the wallet that is due to expire.
sslcertificate.expired	The SSL Certificate expires. All the Autonomous AI Database wallets related to this Autonomous Exadata VM Cluster will expire.
sslcertificate.rotation.reminder	An SSL certificate is older than 365 days and recommends the customer to rotate the certificate. After an SSL certificate crosses 365 days, this reminder is once per week until it is rotated.
sslcertificate.rotated	The SSL certificate is rotated either manually (using the Oracle Cloud Infrastructure console or API) or automatically on its expiry.

✓ Tip

Subscribe to these events using the OCI Notifications Service to receive them whenever they are published. Refer to [Creating a Subscription](#) for further details.

Refer to Events for Autonomous AI Database on Dedicated Exadata Infrastructure for a comprehensive list of Autonomous AI Database events.

Data Protection

Data protection is a critical aspect of data security in any database. Privileged database accounts are one of the most commonly used pathways for gaining access to sensitive applications data in the database. While privileged users such as ADMIN or Oracle operators need broad and unrestricted access to facilitate database maintenance, the same access also creates a point of attack for gaining access to large amounts of data.

Autonomous AI Database lets you implement **Privileged Access Management (PAM)** using:

- **Oracle Database Vault** comes preconfigured and ready to use in Autonomous AI Databases. You can use its powerful security controls to 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. You can configure trusted paths to add additional security controls to authorized data access and database changes. Through the run-time 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. Database Vault secures existing database environments transparently, eliminating costly and time consuming application changes.

Oracle AI Database Vault predominantly addresses the following database security concerns:

- **Administrative privileged account access to application data:** Although the database administrator is the most powerful and trusted user, this administrator does not need access to application data residing within the database.

Oracle AI Database Vault realms around application schemas, sensitive tables, and stored procedures provide controls to prevent privileged accounts from being exploited by intruders and insiders to access sensitive application data. Refer to *How Oracle AI Database Vault Protects Privileged User Accounts* in Oracle Database 19c Administrator's Guide or Oracle Database 26ai Administrator's Guide for more details.
- **Separation of duties for application data access:** Oracle AI Database Vault separation of duty controls can be customized and organizations with limited resources can assign multiple Oracle AI Database Vault responsibilities to the same administrator, but using separate accounts for each separation-of-duty role to minimize damage to the database if any one account is stolen and leveraged. Refer to *How Oracle AI Database Vault Addresses Database Consolidation Concerns* in Oracle Database 19c Administrator's Guide or Oracle Database 26ai Administrator's Guide for more details.

Before using Database Vault, review *What to Expect After You Enable Oracle AI Database Vault* in Oracle Database 19c Administrator's Guide or Oracle Database 26ai Administrator's Guide to understand the impact of configuring and enabling Database Vault.

For information about how to configure and enable Database Vault in your Autonomous AI Database, see *Use Oracle AI Database Vault to Manage Database User Privileges*.

✓ **Tip**

To try out the process of setting up Database Vault, run *Lab 1: Protect Data With Database Vault* in Oracle Autonomous AI Database Dedicated for Security Administrators Workshop.

- PAM is also used to help secure data for customer-authorized use and to help protect data from unauthorized access, which includes preventing access to customer data by Oracle Cloud Operations and support staff members. **Oracle Operations Access Control** ensures the user accounts that Oracle Cloud operations and support staff use to monitor and analyze the performance cannot access data in Autonomous AI Databases. See *Privileged Access Management* for more information.

Sensitive Data Discovery and Data Masking

Identifying sensitive data (such as credit card number, SSN) and masking or redacting them as needed enhances data protection and overall data security.

- Oracle Autonomous AI Database on Dedicated Exadata Infrastructure supports integration with **Oracle Data Safe** service to help you assess and secure your databases. Oracle Data Safe helps you understand the sensitivity of your data, evaluate risks to data, mask sensitive data, implement and monitor security controls, assess user security, monitor user activity, and address data security compliance requirements in your databases.

It provides the following set of features in a single, easy-to-use management console:

- **Security Assessment** helps you assess the security of your database configuration.
- **User Assessment** helps you assess the security of your database users and identify high risk users.

- **Data Discovery** helps you find sensitive data in your database. Data Masking provides a way for you to mask sensitive data so that the data is safe for non-production purposes.
- **Activity Auditing** lets you audit user activity on your database so you can monitor database usage and be alerted of unusual database activities.

For more information about using Data Safe, see *Oracle Data Safe Overview* in *Using Oracle Data Safe*.

To use Oracle Data Safe to identify and protect sensitive and regulated data of your Autonomous AI Database, you must register your database with Data Safe. After registering your database with Data Safe, you can go to the Data Safe console directly from the Details page of your Autonomous AI Database. For more information about registering your database, see *Register or Deregister a Dedicated Autonomous AI Database with Data Safe*.

- When a query issued by an application at run-time returns a result set with sensitive information such as credit card numbers or personal IDs, **Oracle Data Redaction** can help you mask the sensitive details before returning the result-set to the application. You can implement policies for data redaction using the **DBMS_REDACT** PL/SQL Package. Refer to *DBMS_REDACT* in *Oracle Database 19c PL/SQL Packages and Types Reference* or *Oracle Database 26ai PL/SQL Packages and Types Reference*.

Regulatory Compliance Certification

Autonomous AI Database on Dedicated Exadata Infrastructure meets a broad set of international and industry-specific compliance standards, including:

- FedRAMP High—Federal Risk and Authorization Management Program (U.S. Government Regions only)
- HIPAA—Health Insurance Portability and Accountability Act
- ISO/IEC 27001:2013—International Organization for Standardization 27001
- ISO/IEC 27017:2015—Code of Practice for Information Security Controls Based on ISO/IEC 27002 for Cloud Services
- ISO/IEC 27018:2014—Code of Practice for Protection of Personally Identifiable Information (PII) In Public Clouds Acting as PII Processors
- PCI DSS—Payment Card Industry Data Security Standard is a set of requirements intended to ensure that all companies that process, store, or transmit credit card information maintain a secure environment
- SOC 1—System and Organization Controls 1
- SOC 2—System and Organization Controls 2

For more information and a complete list of certifications, see [Oracle Cloud Compliance](#). See [Compliance documents](#) to download copies of the certification documents.

Configuration Management in Autonomous AI Database on Dedicated Exadata Infrastructure

Built on Oracle Cloud Infrastructure (OCI), Autonomous AI Database on Dedicated Exadata Infrastructure provides standard, hardened security configurations so you and your team don't need to spend huge amounts of time and money managing configurations across your Autonomous AI Database fleet.

Security patches and updates are done automatically, so you don't need to worry about keeping security up to date. These capabilities protect your highly sensitive databases and data from costly and potentially disastrous security vulnerabilities and breaches. Refer to Service Maintenance of Autonomous AI Database for more details.

Autonomous Virtual Machine Hardening

Autonomous AI Database Virtual Machine (Autonomous VM) images, also known as client VMs, are security-hardened. As outlined in [Oracle Software Security Assurance](#), their configurations are secured via Oracle software development and assurance practices. Autonomous VMs have suitable anti-virus and anti-malware software configured to detect unauthorized software and malware. Oracle's Asset Endpoint Protection and Configuration Management software, installed on client virtual machines, ensures configuration changes are made only through secure, approved processes. Linux OS audit logs are collected and transferred to a central OCI Security Information and Event Management (SIEM) system for security incident detection and auditing by OCI's security incident Detection And Response Team (DART). Logs are retained for 13 months from the date of generation.

DART is responsible for manning SIEM dashboards, assessing incident alerts and initiating remediation actions on true positives by opening tickets on internal service teams. When a security event demands a customer update, DART works with Global Information Security and service teams to issue a customer update.

All Oracle Autonomous VMs are DISA STIG (Defense Information Systems Agency Security Technical Implementation Guide) compliant and are hardened according to [Oracle Linux Security Technical Implementation Guide](#), which addresses issues around user access controls, open ports, unwanted packages, and daemon configurations, among others. You can find a complete list of Oracle Linux DISA STIG controls [here](#).

Manual access to Autonomous VMs is restricted to a core cloud operations team thoroughly vetted by the company. Operations team members are required to be on the Oracle Cloud Network Attach (a private, secure cloud network) from a company-provided device to access Exadata Infrastructure. Access credentials are dynamically generated in response to valid support tickets. Any configuration change to the Client VMs undergoes strict internal security review and a change management process. All tools, scripts or software are installed or modified only after going through the approved software life cycle and change management process.

Integration with *Operator Access Control service* for both infrastructure and Autonomous VMs further restricts this access and puts access permission and notification in your hands. Operator actions are logged in near real-time and sent to a customer-configured SIEM and the Oracle logging service for download by the customer as desired. You can download the logs to customer SIEM/Storage or archive them indefinitely in OCI object storage. See Operator Access Control service for more details.

OCI security architecture further defines OCI's unique multi-layered gen2 hardware and virtualization security. You can refer to [Oracle Cloud Infrastructure Security Architecture](#) for more details.

Configuration Drift Management

Autonomous AI Database service development and Autonomous VM image build are part of the scope of Oracle corporate security practices. This implementation is carefully controlled in the Oracle Software Security Assurance process, published [here](#).

Autonomous VM image configurations are controlled via code and undergo multiple code reviews and quality assurance (QA) cycles before a configuration change makes it to a production release. Please refer to the [Secure configurations](#) section of the *Oracle Software*

Security Assurance documentation for Oracle's posture and standard practices for securing software configurations.

An Oracle-built agent, Asset Endpoint Protection and Configuration Management (AEP/CM) software is installed on control plane servers to collect and transfer Linux audit logs and Linux Advanced Intrusion Detection Environment (AIDE) logs from infrastructure and Autonomous VM instances. These logs are transferred to an OCI central SIEM for audit purposes. SIEM rules specific to tampering with log files, downloading external content, disabling security tooling, and others generate alerts that DART assesses and responds to as described [Autonomous Virtual Machine Hardening](#).

Autonomous VM instances are secured from direct ssh access except from approved Oracle operators and automation. All operator activity can be monitored via Operator Access Control.

File Integrity and Intrusion Monitoring

Autonomous VMs are configured with a file intrusion and monitoring utility that maintains the count and integrity of files in a specific build. Any change in file count or change in a file checksum is flagged. AIDE and HIDS logs are also collected and sent to OCI SIEM and scanned for threats via the DART process explained in [Autonomous Virtual Machine Hardening](#).

All software artifacts deployed to an AVMC, including tooling, are deployed via a secure change management method employing checksums and digitally signed using SSL certificates. This is called certificate-signed code deployment.

Autonomous VM Vulnerability Scanning and Response

All Autonomous VM images are built using Oracle's secure development practices as documented in [Oracle Software Security Assurance](#). The Corporate Security Solution Assurance Process (CSSAP) is a security review process developed by Oracle's corporate security architecture, Global Information Security (GIS) and Oracle's IT organizations to provide comprehensive information security management review. GIS and CSSAP operate independent of OCI service teams to protect Customer and Oracle's information and software assets. Every service feature with a potential security impact undergoes a CSSAP review and approval process. In addition, quality assurance (QA) testing cycles use appropriate scanning tools to ensure images adhere to STIG, meet service security guidelines, and are ready for CSSAP review.

Forensic tooling on AVMCs plays a prominent role in vulnerability management. Linux audit logs from each Autonomous VM host are uploaded to a central OCI SIEM where alert rules capture and surface potential threats. DART responds to these alerts as explained in [Autonomous Virtual Machine Hardening](#). HIDS and anti-virus logs are also processed similarly. A Common Vulnerabilities and Exposures (CVE) scanner sends its findings to a central automation tool where vulnerability findings are categorized, and tickets are opened for service teams to patch systems on a timescale proportional to the severity of the finding. All CVEs with a score greater than 7 must be patched within 30 days.

You can schedule infrastructure patch bundles comprising Hypervisor, Grid Infrastructure, Storage and Client Operating systems and Firmware quarterly. Database Release Updates and Release Update Revisions may also be scheduled separately each quarter. All patches are staged and applied using cloud automation tooling and Autonomous Cloud Operations, as the specific patch update requires.

Software patch development follows Oracle's secure software development practices, QA testing, and CSSAP reviews as necessary. Separation of duty between patch developers, QA testers, release management and patching operations ensures multiple personnel are involved before a patch makes it to the customer's hardware.

When possible, updates are applied to the running system without downtime using tools like Linux ksplice. If an update requires a component restart, Oracle performs the component restart in a rolling fashion to ensure service availability during the update process. You may schedule patching start times to align with your business SLAs. Patching may be scheduled separately for infrastructure components (GI, OS) and each DBMS home.

Vulnerability Scans and Patching

Autonomous AI Database on Dedicated Exadata Infrastructure performs external and internal vulnerability scans (which include the discovery of end-of-support systems) frequently using commercial vulnerability scanning tools. Identified vulnerabilities are investigated and tracked to resolution by the Cloud Compliance Standard for Vulnerability Management. It uses various technical measures to evaluate and identify updates for third-party and open-source libraries. Authenticated vulnerability scans of systems deployed into the environment, as well as scans of pre-deployment system images, have been implemented to identify such libraries and determine whether security fixes are needed. Corporate policies and business unit procedures govern these programs and evaluate them annually.

Autonomous AI Database uses a mechanism to aggregate security findings from multiple sources (including vulnerability scans) and assign findings to the appropriate service team. This system lets service teams manage their findings and integrate with ticketing systems for automated queuing of remediation work, including notification and automatic escalations as necessary. The system also summarizes remediation work across the organization and drives day-to-day vulnerability management efforts.

[Oracle Software Security Assurance \(OSSA\)](#) defines Oracle's methodology for building security into its products' design, build, testing, and maintenance, whether used on-premises by customers or delivered through Oracle Cloud. Oracle Corporate Security policies (including policies that address threat and vulnerability management) are reviewed annually and updated as needed. At least annually, independent third parties conduct a system penetration test.

To provide the best security posture to all Oracle customers, Oracle fixes significant security vulnerabilities based on the likely risk that they pose to customers. The issues with the most severe risks are fixed first. In general, the fixes for security vulnerabilities are produced in the following order:

- Main code line first—that is the code line being developed for the next major release of the product.
- For each supported version that is vulnerable, create a Critical Patch Update patch and apply the fix in the next patch set if another patch set is planned for that supported version.

Patches and updates are implemented through continuous integration/continuous deployment (CI/CD) tools. Except where dependencies exist across multiple availability domains (for example, updates to domain name services), changes are implemented separately in each region and availability domain. The Oracle Patching and Security Alerts Implementation Policy requires the deployment of the Oracle Critical Patch Update and Security Alert updates and associated recommendations. This policy also includes requirements for remediating vulnerabilities in non-Oracle technology using a risk-based approach. For more information, see [Critical Patch Update and Security Alert programs](#).

Oracle schedules and performs a **monthly infrastructure security maintenance** activity alongside quarterly maintenance. However, these security patches are applied only in those months with critical security updates, including fixes for vulnerabilities with CVSS scores greater than or equal to 7.

- Any Exadata Infrastructure provisioned before Oracle schedules the security maintenance will be eligible for security maintenance.

- The monthly security maintenance process updates database servers to fix critical security vulnerabilities and product issues. They also update storage servers to an Exadata Storage Software image that resolves known security vulnerabilities and product issues.

Oracle Cloud Security Testing Policy

Oracle regularly performs penetration and vulnerability testing and security assessments against the Oracle Cloud infrastructure, platforms, and applications to validate and improve the overall security of Oracle Cloud services. However, Oracle does not assess or test any components (including non-Oracle applications, non-Oracle databases or other non-Oracle software, code or data, as may be applicable) that you manage through or introduce into – including introduction through your development in or creation in - the Oracle Cloud services (the "**Customer Components**").

[Oracle Customer Security Testing Policy](#) describes the security testing activities such as penetration testing and vulnerability scanning that Oracle customers can perform against their Oracle On-Premises Products and Oracle Cloud Services ("Security Tests" or "Security Testing"). It is collectively referred to as "Testing Policy" and included in the Service Specifications for Oracle Cloud Services. This policy does not address or provide any right to conduct testing of any third-party materials included in the Customer Components. Only customers with an Oracle Account with the necessary privileges to file service maintenance requests and who are signed in to the environment that will be the subject of such testing may conduct any vulnerability or penetration testing.

Except as otherwise permitted or restricted in your Oracle Cloud services agreements, your service administrator who has system-level access to your Autonomous AI Database service may run penetration and vulnerability tests for the Customer Components included in your Autonomous AI Database service in accordance with the restrictions outlined in [Security Testing in the Oracle Cloud](#).

You can also refer to the [Security testing FAQ](#) to find answers to questions you may have about your Security Testing.

Endpoint, Malware, and Ransomware Protection

Autonomous VM client machines are built with endpoint, malware, and ransomware protection as discussed below:

- Suitable anti-virus and anti-malware tools are installed and updated regularly.
- ssh / sftp access on the client network is closed.
- Named user accounts on client VMs are extremely limited to required processes only.
- Autonomous VMs are hardened to DISA STIG standards to ensure no unused ports are open or daemons run on the system.
- Oracle operator access is via a secured, outbound connection to Oracle's management network in a customer-chosen OCI region.
- Oracle operator actions may be controlled and audited through Operator Access Control service integration.

Security Incident Management

A dedicated team of security analysts from the security incident Detection and Response Team (DART) is responsible for manning the security event dashboards 24 / 7 and processing alerts to filter true positives. If a true positive is detected, a suitable response is initiated based on the

severity and impact of the event. This can include further analysis, a root cause assessment and fix with the service teams and customer communication.

Oracle's security incident response policies are outlined in [Security Incident Response](#).

Data Encryption in Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database on Dedicated Exadata Infrastructure uses always-on encryption that protects data at rest and in transit. All data stored in and network communication with Oracle Cloud is encrypted by default. Encryption cannot be turned off.

Encryption of Data at Rest

Data at rest is encrypted using TDE (Transparent Data Encryption), a cryptographic solution that protects the processing, transmission, and storage of data. Each Autonomous AI Database on Dedicated Exadata Infrastructure has its own encryption key, and its backups have their own different encryption key.

By default, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure creates and manages all the master encryption keys used to protect your data, storing them in a secure PKCS 12 keystore on the same Exadata systems where the databases reside. If your company security policies require, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure can instead use keys you create and manage in the Oracle Cloud Infrastructure Vault service or Oracle Key Vault, depending on whether you are deploying Oracle Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Cloud or on Exadata Cloud@Customer. For more information, see [Manage Master Encryption Keys](#).

Additionally, regardless of whether you use Oracle-managed or customer-managed keys, you can rotate the keys used in existing databases when needed in order to meet your company security policies.

Note

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

Encryption of Data in Transit

Clients (applications and tools) connect to an Autonomous AI Database using Oracle Net Services (also known as SQL*Net) and predefined database connection services. Oracle Autonomous AI Database on Dedicated Exadata Infrastructure provides two types of database connection services, each with its own technique for encrypting data in transit between the database and the client:

- **TCPS (Secure TCP) database connection services** use the industry-standard TLS 1.2 and TLS 1.3 (Transport Layer Security) protocol for connections. However, TLS 1.3 is only supported on Oracle Database 23ai or later.

When you create an Autonomous AI Database, a connection wallet is generated, containing all necessary files for a client to connect using TCPS. You distribute this wallet only to those clients you want to give database access to, and the client-side configuration uses information in the wallet to perform symmetric-key data encryption.

- **TCP database connection services** use the Native Network Encryption cryptosystem built into Oracle Net Services to negotiate and encrypt data during transmission. For this negotiation, Autonomous AI Databases are configured to require encryption using AES256, AES192 or AES128 cryptography.

Because the encryption is negotiated when the connection is made, TCP connections do not need the connection wallet required for TCPS connections. However, the client does need information about the database connection services. This information is available by clicking **DB Connection** on the database's **Autonomous AI Database Details** page in the Oracle Cloud Infrastructure console and in the `tnsnames.ora` file that is included in the same downloadable zip file containing the files necessary to connect using TCPS.

You can encrypt table data while exporting to Object Storage using `DBMS_CRYPT0` encryption algorithms or a user-defined encryption function. The encrypted data in Object Storage can also be decrypted for use in an external table or while importing from Object Storage using the `DBMS_CRYPT0` encryption algorithms or a user-defined encryption function. See [Encrypt Data While Exporting to Object Storage and Decrypt Data While Importing from Object Storage](#) for instructions.

Auditing Capabilities in Autonomous AI Database

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

Auditing Service Level Activities

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

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

Additionally, when Oracle Autonomous AI Database on Dedicated Exadata Infrastructure performs operations on 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 the rules and actions it uses, see [Overview of Events](#). For listings of the Autonomous AI Database operations that generate events, see [Events for Autonomous AI Database on Dedicated Exadata Infrastructure](#).

✓ Tip

To try out using the Events and Notifications services to create notifications, you can go through *Lab11: OCI Notification Service* in *Oracle Autonomous AI Database Dedicated for Fleet Administrators*.

Auditing Database Activities

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure configures the Autonomous AI Databases you create to use the unified auditing feature of Oracle AI Database.

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

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

Therefore, you can use the unified audit trail to perform a wide variety of diagnostic and security analysis activities on your database.

To keep the unified audit trail from growing too large, Autonomous AI Databases you create include an Oracle Scheduler job named `MAINTAIN_UNIAUD_TRAIL` that runs on a daily basis to remove unified audit records older than 90 days. You can, as the `ADMIN` database user, change this job's characteristics.

As a user with the `AUDIT_ADMIN` role, you can create or modify the audit policies. As a user with the `AUDIT_VIEWER` role, you can view the unified audit data by querying the following views:

- `AUDIT_UNIFIED_CONTEXTS`
- `AUDIT_UNIFIED_ENABLED_POLICIES`
- `AUDIT_UNIFIED_POLICIES`
- `AUDIT_UNIFIED_POLICY_COMMENTS`

Only an `ADMIN` user can grant the `AUDIT_VIEWER` or `AUDIT_ADMIN` roles to another user. Having the `PDB_DBA` role does not allow you to grant `AUDIT_VIEWER` or `AUDIT_ADMIN` to other users.

For more information about how unified auditing works and how to use it, see *What Is Unified Auditing?* in Oracle Database 19c Security Guide or Oracle Database 26ai Security Guide.

Additionally, if you register your Autonomous AI Database with Oracle Data Safe you can use its extensive activity auditing and activity-based alerts features.

For information about these Data Safe features, see Activity Auditing in *Using Oracle Data Safe*. For information about registering your database with Data Safe, see Register or Deregister a Dedicated AI Database with Data Safe.

Auditing Autonomous VM Activities

Collection agents running on the control plane servers of Autonomous AI Database collect and send OS audit logs for all virtual machines and hypervisors running on the physical host, in addition to logs for anti-virus and host intrusion detection software. These logs are sent to a central System Information and Event Management (SIEM) service in OCI. Hundreds of alert rules on the SIEM scan for configuration changes, potential intrusions and unauthorized access attempts, among others.

A dedicated team of security analysts from the security incident Detection and Response Team (DART) is responsible for manning the security event dashboards 24 / 7 and processing alerts to filter true positives. If a true positive is detected, a suitable response is initiated based on the severity and impact of the event. This can include further analysis, a root cause assessment and fix with the service teams and customer communication.

In addition, vulnerability scanning software sends its findings to OCI Security Central, which automatically generates tickets for service teams to resolve findings within a time-period depending on the CVSS score. In addition, audit events for operation actions are sent to the logging service and a customer-provided syslog for systems enrolled in the Operator Access Control Service.

Oracle retains the following logs for Autonomous VMs on Exadata Cloud@Customer X8M and later hardware:

- /var/log/messages
- /var/log/secure
- /var/log/audit/audit.log
- /var/log/clamav/clamav.log
- /var/log/aide/aide.log

Oracle retains the following infrastructure audit logs for Exadata Cloud@Customer X8M and later hardware:

- Integrated Lights-Out Management (ILOM)
 - ILOM syslog redirected to the syslog of the physical infrastructure component
 - syslog
- Physical Exadata Database Server
 - /var/log/messages
 - /var/log/secure
 - /var/log/audit/audit.log
 - /var/log/clamav/clamav.log
 - /var/log/aide/aide.log
- Exadata Storage Server
 - /var/log/messages
 - /var/log/secure
 - /var/log/audit/audit.log

Auditing Oracle Operator Activities

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure provides robust auditing capabilities that extend the existing auditing capabilities to include those activities performed by Oracle operators with a primary focus on enabling regulatory requirements for control and audit of all aspects of system management.

✓ Tip

For step by step guidance on how to create and manage operator control access to Exadata Infrastructure and Autonomous Exadata VM Cluster resources, see *Lab 15: Operator Access Control* in Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop.

Autonomous AI Database on Dedicated Exadata Infrastructure works in a shared responsibility model, where:

- You, the customer are responsible for the data and the database application.
- Oracle is responsible for the infrastructure components: power, bare-metal operating system, hypervisors, Exadata Storage Servers, and other aspects of the infrastructure environment.
- Oracle is responsible for the DBMS software and for the overall health of the database.

In this model, Oracle has unfettered access to the components it is responsible for. This can be a problem if you have regulatory requirements to audit and control all aspects of your system management.

Oracle Operator Access Control is a compliance audit system that enables you to maintain close management and audit trails of all actions that an Oracle operator performs on the Exadata Infrastructure, Exadata Infrastructure hosting an Autonomous AI Database, and Autonomous Exadata VM Cluster (client virtual machines deployed on Oracle Autonomous AI Database) administered by Oracle. In addition, customers can control and limit operator's access to a specific customer-approved Autonomous Container Database (ACD).

Oracle Operator Access Control lets you:

- Control who can access the system, when the system can be accessed, and how long Oracle personnel can access the system.
- Limit access, including limiting what actions an Oracle operator can perform on your system.
- Revoke access, including previously scheduled access that you have granted.
- View and save a near real-time report of all actions an Oracle operator performs on your system.

You can use Oracle Operator Access Control to:

- Control and audit all actions performed by any operator or system software on the following Autonomous AI Database resources:
 - Exadata Infrastructure
 - Autonomous Exadata VM Cluster (AVMC)
 - Autonomous Container Database (ACD)

See Enforcement of Actions in Operator Access Control to know more about enforcing controls on the operations an Oracle operator can perform in your environment.

- Provide controls for client virtual machines deployed on Oracle Autonomous AI Database on Dedicated Exadata Infrastructure. Like Operator access control for Exadata Cloud@Customer Infrastructure, customers can impose Oracle operator access controls on their Autonomous Virtual Machine clusters deployed on Exadata Cloud@Customer or Oracle Public Cloud. See Operator Access Control Actions: Autonomous VM Cluster for further information.
- Maintain the same level of audits and access control to your systems. See How Operator Access is Audited for more information.
- Provide the audit records required for internal or external regulatory audits across your systems. See Managing and Searching Logs with Operator Access Control for more details.

While creating an Operator Control, you can choose either **Exadata Infrastructure** or **Autonomous Exadata VM Cluster** depending on the resource that you want to audit for operator access. See Create Operator Control for more details.

Access Control Within Autonomous AI Database on Dedicated Exadata Infrastructure

When configuring Autonomous AI Database on Dedicated Exadata Infrastructure, you need to ensure that your cloud users have access to use and create only the appropriate kinds of cloud resources to perform their job duties. Additionally, you need to ensure that only authorized personnel and applications have access to the Autonomous AI Databases created on dedicated infrastructure. Otherwise, you run the risk of "runaway" consumption of your dedicated infrastructure resources or inappropriate access to mission-critical data.

Before you begin creating and using the cloud resources that provide the dedicated infrastructure feature, you need to formulate an access control plan, and then institute by creating appropriate IAM (Identity and Access Management) and Networking resources. Accordingly, access control within an Autonomous AI Database is implemented at various levels:

- Oracle cloud user access control
- Client access control
- Database user access control

Oracle Cloud User Access Control

You control what access the Oracle Cloud users in your tenancy have to the cloud resources that make up your deployment of Autonomous AI Database on Dedicated Exadata Infrastructure.

You use the **Identity and Access Management (IAM)** service to ensure that your cloud users have access to create and use only the appropriate kinds of Autonomous AI Database resources to perform their job duties. To institute access controls for cloud users, you define policies that grant specific groups of users specific access rights to specific kinds of resources in specific compartments.

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

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

Policy and Policy Statements

The primary tool you use to define access control for cloud users is the **policy**, an IAM (Identity and Access Management) resource containing policy statements that specify access in terms of "Who", "How", "What" and "Where".

The format of a policy statement is:

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

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

In the context of the dedicated infrastructure feature, a fleet administrator can manage autonomous container databases, while a database administrator can only use them to create Autonomous AI Databases.

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

Because dedicated infrastructure resources use networking resources, some of the policy statements you create will refer to the **virtual-network-family** resource-type value. Also,

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

- **in compartment <compartment-name>** specifies the "Where" by providing the name of an existing IAM compartment, an IAM resource in which resources are created. Compartments are a fundamental component of Oracle Cloud Infrastructure for organizing and isolating cloud resources.

For the policy details for Autonomous AI Database, refer to IAM Policies for Autonomous AI Database on Dedicated Exadata Infrastructure.

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

Best Practices When Planning and Instituting Access Controls

When planning and instituting your access controls for the dedicated infrastructure feature, you should consider these best practices.

- **Create a separate VCN that contains only private subnets.** In almost every case, the Autonomous AI Databases created on dedicated infrastructure house data that is company-sensitive and is normally accessible only from within the company's private network. Even the data shared with partners, suppliers, consumers and customers is made available to them through regulated, secure channels.

Therefore, the network access you provide to such databases should be private to your company. You can ensure this by creating a VCN that uses private subnets and an IPSec VPN or FastConnect to connect to your company's private network. For information about setting up such a configuration, see Scenario B: Private Subnets with a VPN in *Oracle Cloud Infrastructure Documentation*.

For additional information about securing network connectivity to your databases, see *Ways to Secure Your Network* in *Oracle Cloud Infrastructure Documentation*.

- **Create at least two subnets.** You should create at least two subnets: one for Autonomous Exadata VM Cluster and Autonomous Container Database resources and one for resources associated with clients and applications of Autonomous AI Database.
- **Create at least two compartments.** You should create at least two compartments: one for Exadata Infrastructure, Autonomous Exadata VM Cluster and Autonomous Container Database resources and one for Autonomous AI Database resources.
- **Create at least two groups.** You should create at least two groups: one for fleet administrators and one for database administrators.

Client Access Control

Client access control is implemented in Autonomous AI Database by controlling network access control and client connections.

Network Access Control

You define network access control to Autonomous AI Databases when you set up and configure your dedicated deployment of Oracle Autonomous AI Database on Dedicated Exadata Infrastructure. How you do so depends on whether your dedicated deployment is on Oracle Public Cloud or Exadata Cloud@Customer:

- On **Oracle Public Cloud**, you define network access control using components of the Networking service. You create a Virtual Cloud Network (VCN) containing private Subnets

in which your Autonomous AI Databases are network-accessible. Additionally, you create Security Rules to allow a particular type of traffic in or out the IP addresses in a subnet.

For detailed information about creating these resources, run *Lab 1: Prepare Private Network for OCI Implementation* in Oracle Autonomous AI Database Dedicated for Fleet Administrators.

- On **Exadata Cloud@Customer**, you define network access controls by specifying a client network within your data center and recording it in a VM Cluster Network resource within the Exadata Infrastructure resource.

Zero Trust Packet Routing (ZPR)

APPLIES TO: • Oracle Public Cloud only

Oracle Cloud Infrastructure Zero Trust Packet Routing (ZPR) protects sensitive data from unauthorized access through intent-based security policies you write for resources, such as an Autonomous Exadata VM Cluster (AVMC) to which you assign security attributes.

Security attributes are labels that ZPR uses to identify and organize resources. ZPR enforces policy at the network level each time access is requested, regardless of potential network architecture changes or misconfigurations. ZPR is built on the existing network security group (NSG) and security control list (SCL) rules. For a packet to reach a target, it must pass all NSG and SCL rules and ZPR policy. The request is dropped if any NSG, SCL, or ZPR rule or policy doesn't allow traffic.

You can secure networks with ZPR in three steps:

1. Create ZPR artifacts, namely, [security attribute namespaces](#) and [security attributes](#).
2. [Write ZPR policies](#) to connect resources using security attributes. ZPR uses a ZPR Policy Language (ZPL) and enforces restrictions on access to defined resources. As an Autonomous AI Database on Dedicated Exadata Infrastructure customer, you can write ZPL-based policies in your tenancy to ensure that data from **AVMCs** are accessed only by authorized users and resources.
3. [Assign security attributes to resources](#) to enable the ZPR policies.

Note

Avoid entering confidential information when assigning descriptions, tags, security attributes, or friendly names to cloud resources through the Oracle Cloud Infrastructure console, API, or CLI.

See [Getting Started with Zero Trust Packet Routing](#) for more information.

You have the following options to apply ZPR security attributes to an AVMC:

- Apply security attributes when you provision an AVMC. See [Create an Autonomous Exadata VM Cluster](#) for more information.
- Apply security attributes to an existing AVMC resource. See [Configure Zero Trust Packet Routing \(ZPR\) for an AVMC](#) for more information.

As a prerequisite, the following IAM policies must be defined to add ZPR security attributes to an AVMC successfully:

```
allow group <group_name>  
to { ZPR_TAG_NAMESPACE_USE, SECURITY_ATTRIBUTE_NAMESPACE_USE }  
in tenancy
```

```
allow group <group_name>  
to manage autonomous-database-family  
in tenancy
```

```
allow group <group_name>  
to read security-attribute-namespaces  
in tenancy
```

Access Control Lists (ACLs)

For added security, you can enable Access Control Lists (ACLs) in both Oracle Public Cloud and Exadata Cloud@Customer dedicated deployments. An ACL provides additional protection to your database by allowing only the client with specific IP addresses to connect to the database. You can add IP addresses individually, or in CIDR blocks. Both IPv4 and IPv6 based IP's / CIDRs are supported. This allows you to formulate a fine grained access control policy by limiting your Autonomous AI Database's network access to specific applications or clients.

You can optionally create an ACL during the database provisioning, or at any time thereafter. You can also edit an ACL at any time. Enabling an ACL with an empty list of IP addresses makes the database inaccessible. See Set Access Control List for a Dedicated Autonomous AI Database for details.

Note the following about using an ACL with Autonomous AI Database:

- The Autonomous AI Database Service Console is not subject to ACL rules.
- Oracle Application Express (APEX), RESTful services, SQL Developer Web, and Performance Hub are not subject to ACLs.
- While creating an Autonomous AI Database, if setting an ACL fails, then provisioning the database also fails.
- Updating an ACL is allowed only if the database is in the Available state.
- Restoring a database does not overwrite the existing ACLs.
- Cloning a database, full and metadata, will have the same access control settings as the source database. You can make changes as necessary.
- Backup is not subject to ACL rules.
- During an ACL update, all the Autonomous Container Database (CDB) operations are allowed, but Autonomous AI Database operations are not allowed.

Web Application Firewall (WAF)

For advanced network controls beyond Access Control Lists, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure supports using Web Application Firewall (WAF). WAF protects applications from malicious and unwanted internet traffic. WAF can protect any internet facing endpoint, providing consistent rule enforcement across a customer's applications. WAF provides you with the ability to create and manage rules for internet threats including Cross-Site Scripting (XSS), SQL Injection, and other OWASP-defined vulnerabilities.

Access rules can limit based on geography or the signature of the request. See [Getting Started with Web Application Firewall Policies](#) for steps on how to configure WAF.

Client Connection Control

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure implements client connection control with standard TLS 1.2 and TLS 1.3 certificate-based authentication to authenticate a client connection. However, TLS 1.3 is only supported on Oracle Database 23ai or later.

By default, Autonomous AI Database uses self-signed certificates. However, you can install your CA-signed server-side certificate from the Oracle Cloud Infrastructure (OCI) console. To bring your own certificate, you must first create the certificate using the Oracle Cloud Infrastructure (OCI) Certificate Service as demonstrated in [Creating a Certificate](#). These certificates must be signed and must be in the PEM format, that is, their file extension must be .pem, .cer, or .crt only. For more details, refer to Certificate Management in Dedicated Autonomous AI Database.

Database User Access Control

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure configures the databases you create to use the standard user management feature of Oracle AI Database. It creates one administrative user account, ADMIN, that you use to create additional user accounts and to provide access controls for accounts.

Standard user management provides a robust set of features and controls, such as system and object privileges, roles, user profiles and password policies, that enable you to define and implement a secure database user access strategy in most cases. See [Create and Manage Database Users](#) for detailed instructions.

For basic information about standard user management, see *User Accounts in Oracle AI Database Concepts*. For detailed information and guidance, see *Managing Security for Oracle Database Users* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#).

If your database user access strategy demands more controls than are provided by standard user management, you can configure your Autonomous AI Databases to use any of the following tools to meet more rigorous requirements.

Tool	Description
Database Vault	Oracle Database Vault comes preconfigured and ready to use in Autonomous AI Databases. You can use its powerful security controls to restrict access to application data by privileged database users, reducing the risk of insider and outside threats and addressing common compliance requirements. Refer to <i>Data Protection in Security Features of Autonomous AI Database</i> for more details.
Oracle Cloud Infrastructure Identity and Access Management (IAM)	You can configure Autonomous AI Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous AI Database with IAM credentials. Refer to <i>Use Identity and Access Management (IAM) Authentication with Autonomous AI Database</i> for using this option with your database.

Tool	Description
Azure OAuth2 Access Tokens	You can centrally manage Oracle Autonomous AI Database on Dedicated Exadata Infrastructure users in a Microsoft Azure Active Directory (Azure AD) service with the help of Azure OAuth2 access tokens. This type of integration enables the Azure AD user to access an Oracle Autonomous AI Database on Dedicated Exadata Infrastructure instance. Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to get an Azure AD OAuth2 access token to send to the database. For more information about integrating Microsoft Azure Active Directory with your databases, see Authenticate and Authorize Microsoft Azure Active Directory Users for Autonomous AI Database.
Microsoft Active Directory (CMU-AD)	If you use Microsoft Active Directory as a user repository, you can configure your Autonomous AI Databases to authenticate and authorize Microsoft Active Directory users. This integration can enable you to consolidate your user repository while still implementing a rigorous database user access strategy, regardless of whether you use standard user management, Database Vault, Real Application Security or Virtual Private Database. For more information about integrating Microsoft Active Directory with your databases, see Microsoft Active Directory with Autonomous AI Database.
Kerberos	Kerberos is a trusted third-party authentication system that relies on shared secrets. It presumes that the third party is secure, and provides single sign-on capabilities, centralized password storage, database link authentication, and enhanced PC security. It does this through a Kerberos authentication server. Autonomous AI Database support for Kerberos provides the benefits of single sign-on and centralized authentication of Oracle users. For more information, see Authenticate Autonomous AI Database Users with Kerberos.
Kerberos with CMU-AD	Kerberos authentication can be configured on top of CMU-AD to provide CMU-AD Kerberos authentication for Microsoft Active Directory users. To provide CMU-AD Kerberos authentication for Microsoft Active Directory users, you can enable Kerberos authentication on top of CMU-AD by setting type to CMU while enabling external authentication as demonstrated in the example discussed in Enable Kerberos Authentication on Autonomous AI Database.

Tool	Description
Real Application Security and Virtual Private Database	Oracle Real Application Security (RAS) provides a declarative model that enables security policies that encompass not only the business objects being protected but also the principals (users and roles) that have permissions to operate on those business objects. RAS is more secure, scalable, and cost effective than its predecessor, Oracle Virtual Private Database. With Oracle RAS, application users are authenticated in the application-tier as well as in the database. Irrespective of the data access path, the data security policies are enforced in the database kernel based on the end-user native session in the database. The privileges assigned to the user control the type of operations (select, insert, update and delete) that can be performed on rows and columns of the database objects. For more information about Oracle RAS, see <i>Introducing Oracle Database Real Application Security</i> in Oracle Database 19c Real Application Security Administrator's and Developer's Guide or Oracle Database 26ai Real Application Security Administrator's and Developer's Guide. Oracle Autonomous AI Database on Dedicated Exadata Infrastructure also supports Oracle Virtual Private Database (VPD), the predecessor of Oracle RAS. If you are already familiar with and use Oracle VPD, you can configure and use it with you Autonomous AI Databases. For more information about Virtual Private Database, see <i>Using Oracle Virtual Private Database to Control Data Access</i> in Oracle Database 19c Security Guide or Oracle Database 26ai Security Guide.

Privileged Access Management (PAM)

Oracle's security posture around user access and privilege management across its products and services is documented in [Oracle Access Control](#).

Autonomous AI Database on Dedicated Exadata Infrastructure is designed to isolate and protect customer services and database data from unauthorized access. Autonomous AI Database separates duties between the customer and Oracle. The customer controls access to database schema(s). Oracle controls access to Oracle-managed infrastructure and software components.

Autonomous AI Database on Dedicated Exadata Infrastructure is designed to help secure data for customer-authorized use and to help protect data from unauthorized access, which includes preventing access to customer data by Oracle Cloud Ops staff members. Security measures designed to protect against unauthorized access to the Exadata Infrastructure, Autonomous VMs, and Oracle database data include the following:

- Oracle Database data is protected by Oracle Transparent Data Encryption (TDE) keys.
- The customer controls access to TDE encryption keys and may choose to store such keys in an external Oracle Key Vault key management system.

- Oracle Database Vault is pre-configured to prevent privileged users from accessing customer data in Autonomous AI Databases.
- Customers may choose to approve Operator access via Operator Access Control service enrollment.
- All operator access is based on FIPS 140-2 level 3 hardware multi-factor authentication, implemented with a hardware YubiKey implemented with Oracle-approved devices.
- All operator actions are logged at the command level and can be sent to the OCI logging service or a customer SIEM on a near real-time basis.
- Oracle Operations Access Control ensures that the user accounts that Oracle Cloud operations and support staff use to monitor and analyze the performance cannot access data in Autonomous AI Databases. Oracle Cloud operations and support staff do not have access to the data in your Autonomous AI Databases. When you create an Autonomous Container Database, Autonomous AI Database on Dedicated Exadata Infrastructure enables and configures the Operations Control feature of Oracle Database Vault to block common users from accessing data in Autonomous AI Databases created in the container database.

You can confirm that Operations Control is active by entering this SQL statement in an Autonomous AI Database:

```
SELECT * FROM DBA_DV_STATUS;
```

The status of APPLICATION CONTROL indicates that Operations Control is active.

 Note

Operations Control was earlier known as Application Control.

PAM is also implemented with Database Vault for data protection, as discussed in Security Features of Autonomous AI Database.

IAM Policies for Autonomous AI Database on Dedicated Exadata Infrastructure

This article lists the IAM policies required for managing the infrastructure resources of Autonomous AI Database on dedicated Exadata infrastructure.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure relies on the IAM (Identity and Access Management) service to authenticate and authorize cloud users to perform operations that use any of the Oracle Cloud Infrastructure interfaces (the console, REST API, CLI, or SDK). The IAM service uses **groups**, **compartments**, and **policies** to control which cloud users can access which resources.

Policy Details for Autonomous AI Database

This topic covers details for writing policies to control access to Autonomous AI Database resources.

A policy defines what kind of access a group of users has to a specific resource in an individual compartment. For more information, see [Getting Started with Policies](#).

 Tip

For a sample policy, see [Let database and fleet admins manage Autonomous AI Databases](#).

Resource-Types

An aggregate resource-type covers the list of individual resource-types that directly follow. For example, writing one policy to allow a group to have access to the autonomous-database-family is equivalent to writing four separate policies for the group that would grant access to the autonomous-databases, autonomous-backups, autonomous-container-databases, and cloud-autonomous-vmclusters resource-types. For more information, see [Resource-Types](#).

Resource-Types for Autonomous AI Database**Aggregate Resource-Type:**

autonomous-database-family

Individual Resource-Types:

autonomous-databases

autonomous-backups

autonomous-container-databases

cloud-autonomous-vmclusters (*Oracle Public Cloud deployments only*)

autonomous-vmclusters (*Oracle Exadata Cloud@Customer deployments only*)

autonomous-virtual-machine

 Tip

The cloud-exadata-infrastructures and exadata-infrastructures resource-types needed to provision Autonomous AI Database on Oracle Public Cloud and Exadata Cloud@Customer respectively is covered by the aggregate resource-type database-family. For more information about the resources covered by database-family, see Policy Details for Exadata Cloud Service Instances and Policy Details for Base Database Service.

Supported Variables

General variables are supported. See [General Variables for All Requests](#) for more information. Additionally, you can use the `target.workloadType` variable, as shown in the following table:

<code>target.workloadType</code> value	Description
OLTP	Online Transaction Processing, used for Autonomous AI Databases with Autonomous Transaction Processing workload.
DW	Data Warehouse, used for Autonomous AI Databases with Autonomous Data Warehouse workload.

Example policy using the `target.workloadType` variable:

```
Allow group ADB-Admins
to manage autonomous-database
in tenancy where target.workloadType = 'workload_type'
```

Details for Verb + Resource-Type Combinations

The level of access is cumulative as you go from `inspect` > `read` > `use` > `manage`. A plus sign (+) in a table cell indicates incremental access compared to the cell directly above it, whereas "no extra" indicates no incremental access.

For example, the `read` verb for the `autonomous-databases` resource-type covers the same permissions and API operations as the `inspect` verb, plus the `AUTONOMOUS_DATABASE_CONTENT_READ` permission. The `read` verb partially covers the `CreateAutonomousDatabaseBackup` operation, which also needs `manage` permissions for `autonomous-backups`.

The following tables show the Permissions and API operations covered by each verb. For information about permissions, see [Permissions](#).

For autonomous-database-family Resource Types

Note

The resource family covered by **autonomous-database-family** can be used to grant access to database resources associated with all the Autonomous AI Database workload types.

autonomous-databases

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
inspect	<code>AUTONOMOUS_DATABASE_INSPECT</code>	<code>GetAutonomousDatabase</code> , <code>ListAutonomousDatabases</code>	<i>none</i>
read	<i>INSPECT</i> + <code>AUTONOMOUS_DATABASE_CONTENT_READ</code>	<i>no extra</i>	<code>CreateAutonomousDatabaseBackup</code> (also needs <code>manage</code> <code>autonomous-backups</code>)
use	<i>READ</i> + <code>AUTONOMOUS_DATABASE_CONTENT_WRITE</code> <code>AUTONOMOUS_DATABASE_UPDATE</code>	<code>UpdateAutonomousDatabase</code>	<code>RestoreAutonomousDatabase</code> (also needs <code>read</code> <code>autonomous-backups</code>) <code>ChangeAutonomousDatabaseCompartment</code> (also needs <code>read</code> <code>autonomous-backups</code>)
manage	<i>USE</i> + <code>AUTONOMOUS_DATABASE_CREATE</code> <code>AUTONOMOUS_DATABASE_DELETE</code>	<code>CreateAutonomousDatabase</code>	<i>none</i>

autonomous-backups

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
inspect	AUTONOMOUS_DB_BACKUP P_INSPECT	ListAutonomousDatabaseBackups, GetAutonomousDatabaseBackup	<i>none</i>
read	<i>INSPECT</i> + AUTONOMOUS_DB_BACKUP P_CONTENT_READ	<i>no extra</i>	RestoreAutonomousDatabase (also needs use autonomous-databases) ChangeAutonomousDatabaseCompartment (also needs use autonomous-databases)
use	<i>READ</i> + <i>no extra</i>	<i>no extra</i>	<i>none</i>
manage	<i>USE</i> + AUTONOMOUS_DB_BACKUP P_CREATE AUTONOMOUS_DB_BACKUP P_DELETE	DeleteAutonomousDatabaseBackup	CreateAutonomousDatabaseBackup (also needs read autonomous-databases)

autonomous-container-databases

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
inspect	AUTONOMOUS_CONTAINER_DATABASE_INSPECT	ListAutonomousContainerDatabases, GetAutonomousContainerDatabase	<i>none</i>
read	<i>INSPECT</i> + <i>no extra</i>	<i>no extra</i>	<i>none</i>
use	<i>READ</i> + AUTONOMOUS_CONTAINER_DATABASE_UPDATE	UpdateAutonomousContainerDatabase ChangeAutonomousContainerDatabaseCompartment	CreateAutonomousDatabase (also needs manage autonomous-databases)
manage	<i>USE</i> + AUTONOMOUS_CONTAINER_DATABASE_CREATE AUTONOMOUS_CONTAINER_DATABASE_DELETE	<i>no extra</i>	CreateAutonomousContainerDatabase, TerminateAutonomousContainerDatabase (both also need use cloud-autonomous-vmclusters, use cloud-exadata-infrastructure)

cloud-autonomous-vmclusters

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
inspect	CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT	ListCloudAutonomousVmClusters GetCloudAutonomousVmCluster	<i>none</i>
read	<i>INSPECT + no extra</i>	<i>no extra</i>	<i>none</i>
use	<i>READ +</i> CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE	UpdateCloudAutonomousVmCluster ChangeCloudAutonomousVmClusterCompartment	CreateAutonomousDatabase (also needs manage autonomous-databases) CreateAutonomousContainerDatabase (also needs manage autonomous-container-databases)
manage	<i>USE +</i> CLOUD_AUTONOMOUS_VM_CLUSTER_CREATE CLOUD_AUTONOMOUS_VM_CLUSTER_DELETE	<i>no extra</i>	CreateCloudAutonomousVmCluster, DeleteCloudAutonomousVmCluster (both also need use vnics, use subnets, use cloud-exadata-infrastructures)

autonomous-vmclusters

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
inspect	AUTONOMOUS_VM_CLUSTER_INSPECT	ListAutonomousVmClusters GetAutonomousVmCluster	ChangeAutonomousVmClusterCompartment
read	<i>INSPECT + no extra</i>	<i>no extra</i>	<i>none</i>
use	<i>READ +</i> AUTONOMOUS_VM_CLUSTER_UPDATE	ChangeAutonomousVmClusterCompartment	UpdateAutonomousVmCluster CreateAutonomousContainerDatabase TerminateAutonomousContainerDatabase
manage	<i>USE +</i> AUTONOMOUS_VM_CLUSTER_CREATE + AUTONOMOUS_VM_CLUSTER_DELETE	DeleteAutonomousVmCluster	CreateAutonomousVmCluster

autonomous-virtual-machine

Verbs	Permissions	APIs Fully Covered	APIs Partially Covered
inspect	AUTONOMOUS_VIRTUAL_MACHINE_INSPECT	GetAutonomousVirtualMachine ListAutonomousVirtualMachines	<i>none</i>

Permissions Required for Each API Operation

Autonomous Container Database (ACD) and Autonomous AI Database (ADB) are common resources between Oracle Public Cloud, Multicloud, and Exadata Cloud@Customer deployments. Hence, their permissions are the same for both deployments in the following table.

However, certain ACD operations require AVMC-level permissions, and as AVMC resources are different for Oracle Public Cloud and Exadata Cloud@Customer, you need different permissions on each deployment type. For example, to create an ACD, you need:

- AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_CREATE permissions on Exadata Cloud@Customer.
- CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_CREATE permissions on Oracle Public Cloud and Multicloud.

For information about permissions, see [Permissions](#).

The following table lists the API operations for Autonomous AI Database resources in a logical order, grouped by resource type.

Autonomous AI Database API Operations

You can use API to view and manage the different infrastructure resources of an Autonomous AI Database. See API Reference for Autonomous AI Database on Dedicated Exadata Infrastructure for a list of REST API endpoints to manage different Autonomous AI Database resources.

Limiting User Access to Specific Permissions

User access is defined in IAM policy statements. When you create a policy statement giving a group access to a particular verb and resource-type, you're actually giving that group access to one or more predefined IAM permissions. The purposes of verbs is to simplify the process of granting multiple related permissions.

If you want to permit or deny specific IAM permissions, you add a **where** condition to the policy statement. For example, to allow a group of Fleet Administrators to perform any operation on Exadata Infrastructure resources *except* to delete them, you would create this policy statement:

```
Allow group FleetAdmins to manage cloud-exadata-infrastructures in tenancy where
request.permission != 'CLOUD_EXADATA_INFRASTRUCTURE_DELETE'
```

Then, you could allow a smaller group of Fleet Administrators to perform any operation (including deletion) on Exadata Infrastructure resources by omitting the where condition:

```
Allow group FleetSuperAdmins to manage cloud-exadata-infrastructures in tenancy
```

For more information about using the where condition in this way, see the "Scoping Access with Permissions or API Operations" section of [Permissions](#).

Policies to Manage Exadata Infrastructure Resources

The following table lists the IAM policies required for a cloud user to perform management operations on Exadata Infrastructure resources.

Operation	Required IAM Policies on Oracle Public Cloud and Multicloud	Required IAM Policies on Exadata Cloud@Customer
Create an Exadata Infrastructure resource	manage cloud-exadata-infrastructures use vnic use subnet	manage exadata-infrastructures
View a list of Exadata Infrastructure resources	inspect cloud-exadata-infrastructures	inspect exadata-infrastructures
View details of an Exadata Infrastructure resource	inspect cloud-exadata-infrastructures	inspect exadata-infrastructures
Change the maintenance schedule of an Exadata Infrastructure resource	use cloud-exadata-infrastructures	use exadata-infrastructures
Move an Exadata Infrastructure resource to another compartment	use cloud-exadata-infrastructures	use exadata-infrastructures
Manage the security certificates for an Exadata Infrastructure resource	manage cloud-exadata-infrastructures	manage exadata-infrastructures
Terminate an Exadata Infrastructure resource	manage cloud-exadata-infrastructures use vnic use subnet	manage exadata-infrastructures

Policies to Manage Autonomous Exadata VM Clusters

The following table lists the IAM policies required for a cloud user to perform management operations on Autonomous Exadata VM Clusters.

Operation	Required IAM Policies on Oracle Public Cloud and Multicloud	Required IAM Policies on Exadata Cloud@Customer
Create an Autonomous Exadata VM Cluster	manage cloud-autonomous-vmclusters use cloud-exadata-infrastructures	manage autonomous-vmclusters use exadata-infrastructures
View a list of Autonomous Exadata VM Clusters	inspect cloud-autonomous-vmclusters	inspect autonomous-vmclusters
View details of an Autonomous Exadata VM Cluster	inspect cloud-autonomous-vmclusters	inspect autonomous-vmclusters
Change the license type of an Autonomous VM Cluster	Not Applicable	use autonomous-vmclusters inspect exadata-infrastructures

Operation	Required IAM Policies on Oracle Public Cloud and Multicloud	Required IAM Policies on Exadata Cloud@Customer
Move an Autonomous Exadata VM Cluster to another compartment	use cloud-autonomous-vmclusters	use autonomous-vmclusters
Terminate an Autonomous Exadata VM Cluster	manage cloud-autonomous-vmclusters	manage autonomous-vmclusters

Policies to Manage Autonomous Container Databases

The following table lists the IAM policies required for a cloud user to perform management operations on Autonomous Container Databases (ACD).

Operation	Required IAM Policies
Create an Autonomous Container Database	manage autonomous-container-databases use cloud-exadata-infrastructures if creating the Autonomous Container Database on Oracle Public Cloud and Multicloud. use cloud-autonomous-vmclusters if creating the Autonomous Container Database on Oracle Public Cloud and Multicloud. use autonomous-vmclusters if creating the Autonomous Container Database on Exadata Cloud@Customer. use backup-destinations if creating the Autonomous Container Database on Exadata Cloud@Customer.
View a list of Autonomous Container Databases	inspect autonomous-container-databases
View details of an Autonomous Container Database	inspect autonomous-container-databases
Change the backup retention policy of an Autonomous Container Database	use autonomous-container-databases
Edit the maintenance preferences of an Autonomous Container Database	use autonomous-container-databases
Restart an Autonomous Container Database	use autonomous-container-databases
Move an Autonomous Container Database to another compartment	use autonomous-container-databases
Rotate an Autonomous Container Database encryption key	use autonomous-container-databases inspect autonomous-container-databases
Terminate an Autonomous Container Database	manage autonomous-container-databases use cloud-exadata-infrastructures if creating the Autonomous Container Database on Oracle Public Cloud and Multicloud. use cloud-autonomous-vmclusters if creating the Autonomous Container Database on Oracle Public Cloud and Multicloud. use autonomous-vmclusters if creating the Autonomous Container Database on Exadata Cloud@Customer.

Policies to Manage Autonomous Data Guard Configuration

The following table lists the IAM policies required for a cloud user to perform management operations on Autonomous Data Guard configurations.

Operation	Required IAM Policies
View the Autonomous Data Guard Groups with an ACD.	inspect autonomous-container-databases
List the ACDs enabled with Autonomous Data Guard associated with the specified ACD or Autonomous AI Database.	inspect autonomous-container-databases
Reinstate the Disabled Standby to an active standby ACD.	inspect autonomous-container-databases update autonomous-container-databases
Switch Roles of the primary and standby ACDs.	inspect autonomous-container-databases update autonomous-container-databases
Fail Over to the Standby ACD. This standby ACD will become the new primary ACD when the failover completes successfully.	inspect autonomous-container-databases update autonomous-container-databases
Modify Autonomous Data Guard settings such as protection mode, automatic failover, and fast start failover lag limit.	inspect autonomous-container-databases update autonomous-container-databases
Get an Autonomous Data Guard-enabled database associated with the specified Autonomous AI Database.	inspect autonomous-container-databases
List Autonomous AI Database data guard groupss.	inspect autonomous-container-databases
Enable Autonomous Data Guard on an ACD.	inspect cloud-autonomous-vmclusters OR inspect autonomous-vmclusters inspect autonomous-container-databases update autonomous-container-databases
Convert the standby ACD between physical standby and snapshot standby ACD.	inspect cloud-autonomous-vmclusters OR inspect autonomous-vmclusters inspect autonomous-container-databases update autonomous-container-databases

Policies to Manage Autonomous AI Databases

The following table lists the IAM policies required for a cloud user to perform management operations on Autonomous AI Databases.

Operation	Required IAM Policies
Create an Autonomous AI Database	manage autonomous-databases read autonomous-container-databases
View a list of Autonomous AI Databases	inspect autonomous-databases
View details of an Autonomous AI Database	inspect autonomous-databases
Set the password of an Autonomous AI Database's ADMIN user	use autonomous-databases

Operation	Required IAM Policies
Scale the CPU core count or storage of an Autonomous AI Database	use autonomous-databases
Enable or disable auto scaling for an Autonomous AI Database	use autonomous-databases
Move an Autonomous AI Database to another compartment	use autonomous-databases in the Autonomous AI Database's current compartment and in the compartment you are moving it to read autonomous-backups
Stop or start an Autonomous AI Database	use autonomous-databases
Restart an Autonomous AI Database	use autonomous-databases
Back up an Autonomous AI Database manually	read autonomous-databases manage autonomous-backups
Restore an Autonomous AI Database	use autonomous-databases read autonomous-backups
Clone an Autonomous AI Database	manage autonomous-databases read autonomous-container-databases
Terminate an Autonomous AI Database	manage autonomous-databases

Master Encryption Keys in Autonomous AI Database on Dedicated Exadata Infrastructure

By default, Autonomous AI Database on Dedicated Exadata Infrastructure creates and manages all the master encryption keys used to protect your data, storing them in a secure PKCS 12 keystore on the same Exadata systems where the databases reside. These are referred to as **Oracle-managed** encryption keys.

Using Oracle Managed Keys, Oracle is ensuring the complete lifecycle of the keys as Oracle managed meta data. On Exadata Cloud@Customer deployments, access to backups is a shared responsibility and the customer must ensure database environment and backup file accessibility for Oracle APIs to work with internal key management lifecycle operations.

If your company's security policies require it, Autonomous AI Database can use the keys you create and manage using Oracle Key Store instead. For Oracle Public Cloud deployments, you can also use the Oracle Cloud Infrastructure Vault service to create and manage keys. Customers with regulatory compliance to store keys outside Oracle Cloud or any third-party cloud premises can use an External Key Management Service (External KMS).

When you create a customer-managed key using the OCI Vault service, you can also import your own key material (Bring Your Own Key or BYOK) instead of letting the Vault service generate the key material internally.

Caution

As customer-managed keys stored in Oracle Key Vault (OKV) are external to the database host, any configuration change or interruption that makes the OKV inaccessible to the database using its keys makes its data inaccessible.

In Autonomous AI Database on Oracle Database@AWS, you can use AWS Key Management Service (AWS KMS) to manage Master Encryption Keys.

Additionally, regardless of whether you use Oracle-managed or customer-managed keys, you can rotate the keys used in existing databases when needed in order to meet your company security policies. See [Rotate the Encryption Keys](#) for more details.

Before You Begin: Compartment Hierarchy Best Practice

Oracle recommends that you create a compartment hierarchy for your Autonomous AI Database deployment on dedicated infrastructure as follows:

- A "parent" compartment for the entire deployment
- "Child" compartments for each of the various kinds of resources:
 - Autonomous AI Databases
 - Autonomous Container Databases and infrastructure resources (Exadata Infrastructures and Autonomous Exadata VM Clusters)
 - The VCN (Virtual Cloud Network) and its subnets
 - Vaults that contain your customer-managed keys

Following this best practice is especially important when using customer-managed keys because the policy statement you create to grant Autonomous AI Database access to your keys must be added to a policy that is higher in your compartment hierarchy than the compartment containing your vaults and their keys.

Use Customer-Managed Keys in the Vault Service

Before you can use customer-managed keys stored in the Vault service, you must perform a number of preparatory configuration tasks to create a vault and master encryption keys and then make that vault and its keys available to Autonomous AI Database; specifically:

1. Create a vault in the Vault service by following the instructions in [To create a new vault](#) in *Oracle Cloud Infrastructure Documentation*. When following these instructions, Oracle recommends that you create the vault in a compartment created specifically to contain the vaults containing customer-managed keys, as described in [Compartment Hierarchy Best Practice](#).

After creating the vault, you can create at least one master encryption key in the vault by following the instructions in [To create a new master encryption key](#) in *Oracle Cloud Infrastructure Documentation*. When following these instructions, make these choices:

- **Create in Compartment:** Oracle recommends that you create the master encryption key in the same compartment as its vault; that is, the compartment created specifically to contain the vaults containing customer-managed keys.
- **Protection Mode:** Choose an appropriate value from the drop-down list:
 - **HSM** to create a master encryption key that is stored and processed on a hardware security module (HSM).
 - **Software** to create a master encryption key that is stored in a software file system in the Vault service. Software-protected keys are protected at rest using an HSM-based root key. You may export software keys to other key management devices or to a different OCI cloud region. Unlike HSM keys, software-protected keys are free of cost.
- **Key Shape Algorithm:** AES
- **Key Shape Length:** 256 bits

Note

You can also add an encryption key to an existing vault.

2. Use the Networking service to [Create a Service Gateway, a Route Rule and an Egress Security Rule](#) to the VCN (Virtual Cloud Network) and subnets where your Autonomous AI Database resources reside.
3. Use the IAM service to [Create a Dynamic Group](#) identifying your Autonomous AI Database resources and a policy statement granting that dynamic group access to the master encryption keys you created.

Tip

For a "try it out" alternative that demonstrates these instructions, see *Lab 17: Customer Controlled Database Encryption Keys* in Oracle Autonomous AI Database Dedicated for Security Administrators.

After configuring the customer-managed key using the above steps, you can configure it while provisioning an Autonomous Container Database (ACD) or by rotating the existing encryption key from the Details page of ACD or Autonomous AI Database. Autonomous AI Databases provisioned in this ACD will automatically inherit these encryption keys. See [Create an Autonomous Container Database](#) or [Rotate the Encryption Key of an Autonomous Container Database](#) for more details.

If you want to enable Cross-Region Autonomous Data Guard, you first need to replicate the OCI vault to the region where you want to add the standby database. See [Replicating Vaults and Keys](#) for more details.

Note

Virtual vaults created before the cross-region vault replication feature was introduced can't be replicated across regions. Create a new vault and new keys if you have a vault that you need to replicate in another region and replication isn't supported for that vault. However, all private vaults support cross-region replication. See [Virtual vault cross-region replication](#) for details.

Use Bring Your Own Keys (BYOK) in Vault Service

APPLIES TO:  Oracle Public Cloud only

When you create a customer-managed key using the OCI Vault service, you can also import your own key material (Bring Your Own Key or BYOK) instead of letting the Vault service generate the key material internally.

Before you can bring your own keys into the Vault service, you must perform number of preparatory configuration tasks to create a vault and import the master encryption key and then make that vault and its keys available to Autonomous AI Database; specifically:

1. Create a vault in the Vault service by following the instructions in [To create a new vault](#) in *Oracle Cloud Infrastructure Documentation*. When following these instructions, Oracle recommends that you create the vault in a compartment created specifically to contain the vaults containing customer-managed keys, as described in [Compartment Hierarchy Best Practice](#).

After creating the vault, you can create at least one master encryption key in the vault by following the instructions in [To create a new master encryption key](#) in *Oracle Cloud Infrastructure Documentation*. You can also import a customer encryption key into an existing vault. When following these instructions, make these choices:

- **Create in Compartment:** Oracle recommends that you create the master encryption key in the same compartment as its vault; that is, the compartment created specifically to contain the vaults containing customer-managed keys.
- **Protection Mode:** Choose an appropriate value from the drop-down list:
 - **HSM** to create a master encryption key that is stored and processed on a hardware security module (HSM).
 - **Software** to create a master encryption key that is stored in a software file system in the Vault service. Software-protected keys are protected at rest using an HSM-based root key. You may export software keys to other key management devices or to a different OCI cloud region. Unlike HSM keys, software-protected keys are free of cost.
- **Key Shape Algorithm:** AES
- **Key Shape Length:** 256 bits
- **Import External Key:** To use a customer encryption key (BYOK), select **Import External Key** and provide the following details:
 - Wrapping Key Information. This section is read-only, but you can view the public wrapping key details.
 - Wrapping Algorithm. Select a wrapping algorithm from the drop-down list.
 - External Key Data Source. Upload the file that contains the wrapped RSA key material.

Note

You can either import the key material as a new external key version or click the name of an existing master encryption key and rotate it to a new key version. Refer to [Importing Key Material as an External Key Version](#) for more details.

2. Use the Networking service to [Create a Service Gateway, a Route Rule and an Egress Security Rule](#) to the VCN (Virtual Cloud Network) and subnets where your Autonomous AI Database resources reside.
3. Use the IAM service to [Create a Dynamic Group](#) identifying your Autonomous AI Database resources and a policy statement granting that dynamic group access to the master encryption keys you created.

After configuring the customer-managed BYOK using the above steps, you can use it by rotating the existing encryption key from the **Details** page of Autonomous Container Database or Autonomous AI Database. See [Rotate the Encryption Key of an Autonomous Container Database](#) for more details.

Use External Keys from OCI External Key Management Service (OCI EKMS)

APPLIES TO:  Oracle Public Cloud only

Before you can use external keys from OCI EKMS, you must perform number of preparatory configuration tasks to create a vault and then make that vault and its keys available to Autonomous AI Database.

In OCI EKMS, you can store and control master encryption keys (as external keys) on a third-party key management system hosted outside OCI. You can use this for enhanced data security or if you have regulatory compliance to store keys outside Oracle Public Cloud or any third-party cloud premises. With the actual keys residing in the third-party key management system, you create only key references in OCI.

1. You can create and manage a vault that holds key references in OCI EKMS. You can use the keys from OCI EKMS with Autonomous AI Database on Dedicated Exadata Infrastructure deployed on Oracle Public Cloud. See [Creating a vault in OCI EKMS](#) for more details. When following these instructions, make these choices:
 - **Create in Compartment:** Select a compartment for the OCI EKMS vault.
 - **IDCS Account Name URL:** Enter the authentication URL that you use to access the KMS service. The Console redirects to a sign in screen.
 - **Key Management Vendor:** Select a third-party vendor that deploys key management service. For now, OCI KMS supports only **Thales** as the external key management vendor.
 - **Client application ID:** Enter the OCI KMS client ID generated when you register the confidential client application in Oracle Identity Domain.
 - **Client application secret:** Enter the Secret ID of the confidential client application registered in the Oracle Identity Domain.
 - **Private endpoint in compartment:** Select the private endpoint GUID of the external key management.
 - **External Vault URL:** Enter the vault URL that was generated when you created vault in external key management.
2. Use the Networking service to [Create a Service Gateway, a Route Rule and an Egress Security Rule](#) to the VCN (Virtual Cloud Network) and subnets where your Autonomous AI Database resources reside.
3. Use the IAM service to [Create a Dynamic Group](#) identifying your Autonomous AI Database resources and a policy statement granting that dynamic group access to the master encryption keys you created.

Use Customer-Managed Keys in AWS KMS

In Autonomous AI Database on Oracle Database@AWS, you can use AWS Key Management Service (AWS KMS) to manage Master Encryption Keys.

Before you can use customer-managed keys in the AWS KMS, you must enable AWS key management at the AVMC level. After the AVMC is created, you have the option to enable AWS KMS from the AVMC Details page. As long as the AVMC is empty and has no ACDs in it, you will have the option to disable the AWS KMS key management configuration from the AVMC details page. During the ACD creation, you can choose an AWS KMS key as the Key type.

Create a Service Gateway, a Route Rule and an Egress Security Rule

Oracle Cloud Infrastructure (OCI) Service Gateway provides private, secure access to multiple Oracle Cloud services simultaneously from within a virtual cloud network (VCN) or on-premises network via a single gateway without traversing the internet.

Create a service gateway in the VCN (Virtual Cloud Network) where your Autonomous AI Database resources reside by following the instructions in [Task 1: Create the service gateway](#) in *Oracle Cloud Infrastructure Documentation*.

After creating the service gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to access the Vault service:

1. Go to the **Subnet Details** page for the subnet.
2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** All IAD Services In Oracle Services Network
 - **Target Type:** Service Gateway
 - **Target:** The name of the service gateway you just created in the VCN

If such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.

4. Return to the **Subnet Details** page for the subnet.
5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
6. In the side menu, under **Resources**, click **Egress Rules**.
7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Stateless:** No
 - **Destination:** All IAD Services In Oracle Services Network
 - **IP Protocol:** TCP
 - **Source Port Range:** All
 - **Destination Port Range:** 443

If such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Create a Dynamic Group and a Policy Statement

To grant your Autonomous AI Database resources permission to access customer-managed keys, you create an IAM dynamic group that identifies these resources and then create an IAM policy that grants this dynamic group access to the master encryption keys you created in the Vault service.

When defining the dynamic group, you identify your Autonomous AI Database resources by specifying the OCID of the compartment containing your Exadata Infrastructure resource.

1. Copy the OCID of the compartment containing your Exadata Infrastructure resource. You can find this OCID on the **Compartment Details** page of the compartment.

2. Create a dynamic group by following the instructions in [To create a dynamic group](#) in *Oracle Cloud Infrastructure Documentation*. When following these instructions, enter a matching rule of this format:

```
ALL {resource.compartment.id ='<compartment-ocid>'}
```

where **<compartment-ocid>** is the OCID of the compartment containing your Autonomous Exadata VM Cluster resource.

After creating the dynamic group, navigate to (or create) an IAM policy in a compartment higher up in your compartment hierarchy than the compartment containing your vaults and keys. Then, add a policy statement of this format:

```
allow dynamic-group <dynamic-group-name>
to manage keys
in compartment <vaults-and-keys-compartment>
where all {
target.key.id='<key-ocid>',
request.permission!='KEY_MOVE',
request.permission!='KEY_IMPORT'
}
```

If you are using a replicated virtual vault or replicated virtual private vault for the Autonomous Data Guard deployment, add an additional policy statement of this format:

```
allow dynamic-group <dynamic-group>
to read vaults
in tenancy | compartment <vaults-and-keys-compartment>
```

where **<dynamic-group>** is the name of the dynamic group you created and **<vaults-and-keys-compartment>** is the name of the compartment in which you created your vaults and master encryption keys.

Use Customer-Managed Keys in Oracle Key Vault

Oracle Key Vault (OKV) is a full-stack, security-hardened software appliance built to centralize the management of keys and security objects within your enterprise. You integrate your on-premises OKV deployment with Oracle Autonomous AI Database on Dedicated Exadata Infrastructure to create and manage your own master keys.

Before you can use customer-managed keys stored in OKV, you must perform a number of preparatory configuration tasks as described in [Prepare to Use Oracle Key Vault](#).

After completing the preparatory configuration tasks, you can associate the customer manager keys in OKV while provisioning an Autonomous Container Database (ACD) and all the Autonomous AI Databases provisioned in this ACD will automatically inherit these encryption keys. See [Create an Autonomous Container Database](#) for more details.

Note

If you want to enable Cross-Region Autonomous Data Guard, ensure that you have added Connection IP addresses for your OKV Cluster in the Key Store.

Overview of OKV Endpoints:

Oracle Key Vault endpoints are computer systems like database or application servers, where keys and credentials are used to access data. You must register and enroll an endpoint to communicate with Oracle Key Vault. Afterward, keys in the endpoint can be uploaded to Oracle Key Vault and be shared with other endpoints and then downloaded from these endpoints so that users can access their data.

Only a user with the *System Administrator* role or the *Create Endpoint* privilege can add an endpoint to Oracle Key Vault. After the endpoint is added, the endpoint administrator can enroll the endpoint by downloading and installing the endpoint software at the endpoint. The endpoint can then use the utilities packaged with the endpoint software to upload and download security objects to and from Oracle Key Vault.

Features for Experienced Database Users

Oracle Database Features in Autonomous AI Database on Dedicated Exadata Infrastructure

This article provides information on using Oracle AI Database features and options in Autonomous AI Database on Dedicated Exadata Infrastructure.

For equivalent information in Autonomous AI Database Serverless deployments, see *Autonomous AI Database for Experienced Database Users*.

You can provision an Autonomous AI Database with Oracle Database 19c or Oracle AI Database 26ai, depending on the database software version of its parent Autonomous Container Database (ACD). For example, to create an Autonomous AI Database with Oracle AI Database 26ai, you must choose an ACD whose Oracle database software version is 26ai. For a comprehensive list of database features supported by each of these database versions, see [Oracle Database 26ai](#) or [Oracle Database 19c](#) on Oracle Help Center.

Autonomous AI Database 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.

Let us discuss how are some of the key features of the Oracle database handled in an Autonomous AI Database:

- **Data and temporary tablespaces:** The default data and temporary tablespaces for the database are configured automatically. The name of the default data tablespace is DATA.
- **Database character set:** The database character set is Unicode AL32UTF8.
- **Stored data encryption:** Stored data is encrypted using the AES256 (Advanced Encryption Standard 256-bit cipher key) algorithm.

Note

Autonomous AI Databases created before September 2021 use AES128 algorithm by default.

If needed, you can change the encryption algorithm for a tablespace using the ALTER TABLESPACE command. For example, to change the encryption algorithm of the DATA tablespace to AES256, enter:

```
ALTER TABLESPACE data ENCRYPTION ONLINE USING 'AES256' REKEY;
```

- **Data compression:** For Autonomous AI Transaction Processing workloads, compression is *not* enabled by default. For Autonomous AI Lakehouse workloads, *hybrid columnar compression* is enabled by default for all the tables. To enable or disable compression or to specify different compression methods for your tables using the `table_compression` clause in your CREATE TABLE or ALTER TABLE command.
- **Database access:** You do not have direct access to the database node or the local file system, and SYSTEM or SYSAUX tablespaces.
- **Parallel execution:** By default, parallel execution (parallelism within a SQL statement) is not enabled for the Autonomous AI Transaction Processing workloads. However, for the Autonomous AI Lakehouse workloads, parallel execution is enabled by default, and the degree of parallelism for SQL statements is set based on the number of CPUs in the system and the database service you use when connecting to the database.
 - You can modify the degree of parallelism for a table or an index using the `parallel_clause` or by using a hint. For example, you can disable parallel DML in your session using the following SQL command:

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

See *Oracle Database 19c VLDB and Partitioning Guide* or [Oracle Database 26ai 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 unknown to the end-user. Change the *parallel_clause* value to NOPARALLEL or set the PARALLEL attribute to 1 to specify serial execution:

```
ALTER INDEX index_name NOPARALLEL;
```

or

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

Latest Autonomous AI Database Features

Autonomous AI Database includes the latest Oracle Database features.

Autonomous AI Database 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 19c Administrator's Guide or [Oracle Database 26ai 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 19c SQL Tuning Guide or [Oracle Database 26ai 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 19c SQL Tuning Guide or [Oracle Database 26ai 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 19c Administrator's Guide or [Oracle Database 26ai 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 19c Performance Tuning Guide or [Oracle Database 26ai Performance Tuning Guide](#).

The ADMIN User and the SYS User

In Oracle Autonomous AI Database on Dedicated Exadata Infrastructure 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 AI Database on Dedicated Exadata Infrastructure 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
```

All system privileges with the **ANY** keyword (such as `SELECT ANY TABLE`, `CREATE ANY PROCEDURE`) honor the `COMMON_SCHEMA_ACCESS` lockdown, and an ADMIN user can not use it against the Common user schemas.

Unlike Oracle database on-premises, where the **ANY** keyword applies to all users except **SYS**, the **ANY** privilege only works against non-Common Users in Autonomous AI Database.

✓ **Tip**

To find a list of common users, you can run the following query as an ADMIN user:

```
select username  
from dba_users  
where common = 'YES'  
order by username;
```

✓ **Tip**

To view a list of privileges that can be used with the ANY keyword, you can run the following query as an ADMIN user:

```
select distinct(privilege)  
from dba_sys_privs  
where grantee like 'ADMIN' and privilege like '%ANY%'  
order by privilege;
```

Database Initialization Parameters

Autonomous AI Database configures database initialization parameters automatically when you provision a database. You do not need to set any initialization parameters to start using your service. However, you can modify some parameters if you need to.

List of Initialization Parameters that can be Modified

📘 **Note**

Click an initialization parameter in the following list to learn more about it. For the parameters that are not clickable below, see *Oracle Database 19c Reference* or [Oracle Database 26ai Reference](#) for more information.

```
ALLOW_ROWID_COLUMN_TYPE  
APPROX_FOR_AGGREGATION  
APPROX_FOR_COUNT_DISTINCT  
APPROX_FOR_PERCENTILE
```


AWR_PDB_AUTOFLUSH_ENABLED
CONTAINER_DATA
CURRENT_SCHEMA (Session only by using ALTER SESSION)
CURSOR_SHARING
DB_BLOCK_CHECKING
DDL_LOCK_TIMEOUT
FIXED_DATE
GLOBAL_NAMES
HEAT_MAP
IGNORE_SESSION_SET_PARAM_ERRORS
INMEMORY_OPTIMIZED_ARITHMETIC (Allowed only with ALTER SYSTEM)
INMEMORY_QUERY (Allowed with ALTER SYSTEM and ALTER SESSION)
JOB_QUEUE_PROCESSES (You can only lower its value or bring it back to the original value)
LDAP_DIRECTORY_ACCESS
MAX_IDLE_TIME
NLS_CALENDAR
NLS_COMP
NLS_CURRENCY
NLS_DATE_FORMAT
NLS_DATE_LANGUAGE
NLS_DUAL_CURRENCY
NLS_ISO_CURRENCY
NLS_LANGUAGE
NLS_LENGTH_SEMANTICS
NLS_NCHAR_CONV_EXCP
NLS_NUMERIC_CHARACTERS
NLS_SORT
NLS_TERRITORY
NLS_TIME_FORMAT
NLS_TIME_TZ_FORMAT
NLS_TIMESTAMP_FORMAT
NLS_TIMESTAMP_TZ_FORMAT
OPEN_CURSORS (Allowed range of values are 1000 to 4000)
OPEN_LINKS (You must set SCOPE=SPFILE and restart the Autonomous Database after modifying this parameter)
OPTIMIZER_CAPTURE_SQL_PLAN_BASELINES (Session only by using ALTER SESSION)
OPTIMIZER_IGNORE_HINTS
OPTIMIZER_IGNORE_PARALLEL_HINTS
OPTIMIZER_INMEMORY_AWARE (Allowed with ALTER SYSTEM and ALTER SESSION)
OPTIMIZER_MODE
PARALLEL_MIN_DEGREE
PARALLEL_DEGREE_LIMIT
PLSCOPE_SETTINGS
PLSQL_CCFLAGS
PLSQL_DEBUG
PLSQL_OPTIMIZE_LEVEL
PLSQL_WARNINGS
QUERY_REWRITE_INTEGRITY
RECYCLEBIN
RESULT_CACHE_MODE
SESSION_EXIT_ON_PACKAGE_STATE_ERROR
SQL_TRACE (Allowed only with ALTER SESSION)
STATISTICS_LEVEL (Session only by using ALTER SESSION)
[SYSDATE AT DBTIMEZONE Select a Time Zone for SYSDATE on Autonomous AI Database](#)

TIME_ZONE
UNDO_RETENTION (min=300 and max=86400)

SYSDATE_AT_DBTIMEZONE Select a Time Zone for SYSDATE on Autonomous AI Database

SYSDATE_AT_DBTIMEZONE enables special handling in a session for the date and time value returned in calls to SYSDATE and SYSTIMESTAMP.

Depending on the value of SYSDATE_AT_DBTIMEZONE, you see either the date and time based on the default Autonomous AI Database time zone, Coordinated Universal Time (UTC), or based on the time zone that you set in your database.

Property	Description
Parameter type	Boolean
Default Value	FALSE
Modifiable	ALTER SESSION, ALTER SYSTEM
Range of values	TRUE FALSE

Default Autonomous AI Database Time Zone

The default Autonomous AI Database time zone is Coordinated Universal Time (UTC) and by default calls to SYSDATE and SYSTIMESTAMP return the date and time in UTC.

In order to change database time zone, you can run the following statement.

This example sets the database time zone to UTC-5.

```
ALTER DATABASE SET TIME_ZONE='-05:00';
```

This example sets the database time zone for the US east coast: America/New_York and the time is automatically adjusted at DST switches:

```
ALTER DATABASE SET TIME_ZONE='America/New_York';
```

Note

You must restart the Autonomous AI Database instance for the change to take effect.

After you set the database time zone, by default SYSDATE and SYSTIMESTAMP continue to return date and time in UTC (SYSDATE_AT_DBTIMEZONE is FALSE by default). If you set SYSDATE_AT_DBTIMEZONE to TRUE in a session, SYSDATE and SYSTIMESTAMP return the database time zone.

See *Setting the Database Time Zone* in Oracle Database 19c or [Oracle Database 26ai](#) for more information on using the SET TIME_ZONE clause with ALTER DATABASE.

Using SYSDATE_AT_DBTIMEZONE in a Session

When SYSDATE_AT_DBTIMEZONE is FALSE in a session, calls to SYSDATE and SYSTIMESTAMP return values based on the default Autonomous AI Database time zone, Coordinated Universal Time (UTC).

When `SYSDATE_AT_DBTIMEZONE` is `TRUE` in a session, calls to `SYSDATE` or `SYSTIMESTAMP` return the date and time based on the database time zone.

Note

Setting `SYSDATE_AT_DBTIMEZONE` to `TRUE` only affects the use of `SYSDATE` and `SYSTIMESTAMP` as operators in application SQL (for example, in queries, DML, and CTAS operations). When using this parameter, it is recommended that your client/session timezone matches your database timezone.

Example

The following example returns dates and times for two different time zones, based on the `SYSDATE_AT_DBTIMEZONE` parameter value:

```
SQL> SELECT DBTIMEZONE FROM DUAL;
```

```
DBTIMEZONE
```

```
-----  
-05:00
```

```
SQL> ALTER SESSION SET SYSDATE_AT_DBTIMEZONE=FALSE;
```

```
Session altered.
```

```
SQL> SELECT SYSTIMESTAMP FROM DUAL;
```

```
SYSTIMESTAMP
```

```
-----  
27-JAN-22 06.59.45.708082000 PM GMT
```

```
SQL> ALTER SESSION SET SYSDATE_AT_DBTIMEZONE=TRUE;
```

```
Session altered.
```

```
SQL> SELECT SYSTIMESTAMP FROM DUAL;
```

```
SYSTIMESTAMP
```

```
-----  
27-JAN-22 02.14.47.578946000 PM -05:00
```

Note

When a SYSDATE or SYSTIMESTAMP query is executed in SQL Worksheet of Database Actions, the time and date value that is returned is in UTC (when SYSDATE_AT_DBTIMEZONE parameter is set to TRUE or FALSE). To obtain the database time zone when working in Database Actions, use TO_CHAR() as follows:

```
SQL> SELECT TO_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD"T"HH24:MI:SS TZH": "TzM' )
FROM DUAL;
```

```
TO_CHAR(SYSTIMESTAMP, 'YYYY-MM-DD"T"HH24:MI:SS TZH": "TzM' )
```

```
2022-01-27T14:15:00 -05:00
```

Oracle Database Features with Limited Support in Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database on Dedicated Exadata Infrastructure supports most of the features present in Oracle Database Enterprise Edition. To ensure the security and the performance of Autonomous AI Database, some features have only limited support.

The following Oracle Database features, options and packs have limited support in Autonomous AI Database.

Feature, Option or Pack	Limitations
Database In-Memory	See Database In-Memory .
Fast Ingest	See Fast Ingest.
Index-organized tables (IOTs)	Index-organized tables are not supported, but attempting to create one does not generate an error. Instead, a heap-organized table with a primary key index is created. Therefore, if you use index-organized tables, you should test the applications that use index-organized tables to confirm that they work using heap-organized tables with a primary key indexes.
Java in Oracle Database (sometimes called OJVM or Database Embedded JVM)	Database sessions that use Java in Oracle Database will experience reduced database-side performance when service maintenance operations are being performed on the database.
LogMiner	See Oracle LogMiner .
Oracle Data Pump	Transportable tablespace mode (the TRANSPORT_TABLESPACES parameter) is not supported.
Oracle Database Advanced Queuing	Only the PL/SQL Advanced Queuing APIs are supported. Oracle Call Interface (OCI) notifications for Advanced Queuing are not supported.
Oracle Flashback and Restore Points	See Oracle Flashback and Restore Points .
Oracle Label Security	You cannot specify labeling functions in policies.

Feature, Option or Pack	Limitations
Oracle Machine Learning (OML)	Supports: • OML4SQL on all database versions. • Oracle Data Miner. For OML product documentation, refer to Oracle Machine Learning on Oracle Help Center.
Oracle Real Application Testing	<i>SQL Performance Analyzer</i> and <i>Database Replay</i> are the key components of Oracle Real Application Testing (RAT). Depending on the nature and impact of the system change being tested and the type of test system, you can use either or both components to perform your testing. • Database Replay: See Database Replay for more details. • SQL Performance Analyzer (SPA): See <i>Testing Guide</i> in Oracle Database 19c or Oracle Database 26ai for more information on SPA.
 Tip You can use the SPA features from Enterprise Manager (EM). For more details, see Monitoring and Administration Tasks in <i>Administrator's Guide for Oracle Autonomous AI Databases</i>	
Oracle Text	See Oracle Text .
Oracle XML DB	See Oracle XML DB .

SQL Commands

Autonomous AI Database on Dedicated Exadata Infrastructure allows most of the SQL commands available in Oracle Database. To ensure the security and the performance of Autonomous AI Database, 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 AI Database on Dedicated Exadata Infrastructure. Most of the standard SQL and PL/SQL syntax and constructs used with Oracle Database work in Autonomous AI Database.

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 AI Database.

The following SQL commands and PL/SQL packages are not supported in Autonomous AI Databases:

- ADMINISTER KEY MANAGEMENT

- SYS.DBMS_SHARED_POOL
- CREATE_LIBRARY

SQL Commands with Restrictions in Autonomous AI Database

The following DDL commands are available in Autonomous AI Database with restrictions:

SQL Command	Restrictions
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
ALTER PROFILE	Using ALTER PROFILE: • There are restrictions for a user defined PASSWORD_VERIFY_FUNCTION. See Manage Password Complexity on Autonomous AI Database for more information. • The optional CONTAINER clause is ignored if specified.
ALTER SESSION	Only the following clauses are allowed: ADVISE COMMIT, ADVISE ROLLBACK, ADVISE NOTHING CLOSE DATABASE LINK ENABLE COMMIT IN PROCEDURE, DISABLE COMMIT IN PROCEDURE ENABLE PARALLEL <QUERY DDL DML>, DISABLE PARALLEL <QUERY DDL DML>, FORCE PARALLEL <QUERY DDL DML> ENABLE RESUMABLE, DISABLE RESUMABLE SET DEFAULT_COLLATION SET EDITION SET ISOLATION_LEVEL SET ROW ARCHIVAL VISIBILITY SET <i>init-param</i> = <i>value</i> For a list of initialization parameters you can set in Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, see Modifying Database Initialization Parameters. Note that you cannot set some of these parameters at the session level. To find out whether a given parameter can be altered using ALTER SESSION SET, query the ISSUES_MODIFIABLE column of the V\$PARAMETER dynamic performance view.
ALTER SYSTEM	Only ALTER SYSTEM SET and ALTER SYSTEM KILL SESSION are allowed. SET can only be used to set parameters listed in Modifying Database Initialization Parameters.
ALTER TABLESPACE	The use of SMALLFILE tablespaces is not supported by Autonomous AI Database. The Autonomous AI Database block size is fixed at 8K. A BIGFILE tablespace contains only one data file or temp file, and the maximum size of the file for a tablespace with 8K blocks is 32 terabytes (TB).

SQL Command	Restrictions
ALTER TABLE	The following clauses are ignored: - allocate_extent - alter_iot - deallocate_unused - ilm - inmemory_table - logging - modify_LOB_storage - physical_attributes - shrink For more information on ALTER TABLE, see <i>Oracle Database 19c SQL Language Reference</i> or Oracle AI Database 26ai SQL Language Reference.
CREATE DATABASE LINK	In the USING 'connect string' clause, you must use the Easy Connect syntax or provide a complete descriptor. You cannot use a network service name because the tnsnames.ora file is not available for lookup. Database links can only be used for TCP connections because TCPS connections require a wallet.
CREATE PROFILE	Refer to Manage Password Complexity on Autonomous AI Database for the Password Verify Function requirements.
CREATE TABLE	XMLType tables using XML schema-based storage are not allowed. See Oracle XML DB. The following clauses are ignored: - cluster - ilm_clause - inmemory_table_clause - LOB_storage_clause - logging_clause - organization external: Use of this clause has certain limitations, as described in Manually Created External Tables. - organization index: Using this clause does not create an index-organized table. It creates a regular table with a primary key. You should test and verify the performance of the generated table for your application. - physical_properties For more information on CREATE TABLE, see <i>Oracle Database 19c SQL Language Reference</i> or Oracle AI Database 26ai SQL Language Reference.
CREATE TABLESPACE	The use of SMALLFILE tablespaces is not supported by Autonomous AI Database. The Autonomous AI Database block size is fixed at 8K. A BIGFILE tablespace contains only one data file or temp file, and the maximum size of the file for a tablespace with 8K blocks is 32 terabytes (TB).
CREATE USER	The following clauses can not be used with CREATE USER statement: - IDENTIFIED EXTERNALLY - IDENTIFIED BY VALUES

SQL Command	Restrictions
DROP TABLESPACE	This command is supported only when used in the following format: DROP TABLESPACE <tablespace_name> INCLUDING CONTENTS AND DATAFILES; If you execute this command without INCLUDING CONTENTS AND DATAFILES in the syntax, you will get the following error: ORA-01031: insufficient privileges

Data Types

Autonomous AI Database on Dedicated Exadata Infrastructure allows most of the data types available in Oracle Database. To ensure the security and the performance of Autonomous AI Database, some data types are restricted.

The following Oracle Database data types are not supported or are only partially supported in Autonomous AI Database:

- By default, the ROWID and UROWID data types are not enabled in Dedicated Exadata Infrastructure deployments. You cannot specify ROWID or UROWID columns when creating tables, nor can you add ROWID or UROWID columns to existing tables.

However, you can enable ROWID or UROWID by setting the ALLOW_ROWID_COLUMN_TYPE initialization parameter to true. See Database 19c Reference or [Database 26ai Reference](#) for more information on enabling it.

- 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 19c SQL Language Reference* or [Oracle Database 26ai SQL Language Reference](#).

Database In-Memory

Oracle Database In-Memory (Database In-Memory) is a suite of features that significantly improves performance for real-time analytics and mixed workloads.

The In-Memory Column Store (IM column store) is the key feature of Database In-Memory. The IM column store maintains copies of tables, partitions, and individual columns in a special compressed columnar format optimized for rapid scans. The IM column store resides in the In-Memory Area, which is an optional portion of the system global area (SGA). For more information about this feature, refer to Database 19c In-Memory Guide or [Database 26ai In-Memory Guide](#).

Autonomous AI Database on Dedicated Exadata Infrastructure lets you enable Database In-memory for your Autonomous AI Database by allocating a percentage of its System Global Area (SGA) to the In-Memory column store (IM column store) either while provisioning the database or later. You can also enable or disable Database In-Memory for an existing Autonomous AI Database from its **Details** page on the Oracle Cloud Infrastructure (OCI) console.

The following describes notes to consider before and after enabling Database In-Memory for your Autonomous AI Database:

- Turning Database In-Memory on or off is supported only on Autonomous AI Databases with at least 16 ECPU or 4 OCPU enabled.

- Scaling down an Autonomous AI Database to less than 16 ECPUs or 4 OCPUs automatically disables Database In-Memory for that database.
- You can also enable Database In-Memory on databases cloned from a database instance or backup, irrespective of whether the clone source has Database In-memory enabled. As the settings are not inherited from the clone, you can enable Database In-Memory manually while cloning or later.
- You may experience performance degradation in the following cases:
 - Any patching activity that results in rebooting the database with Database In-Memory enabled.
 - Restarting an Autonomous Container Database with Database In-Memory enabled Autonomous AI Databases.
- If you enable Database In-Memory for a primary Autonomous AI Databases in an Autonomous Data Guard setup, the standby database also carries the same settings. However, you can not change these settings on the standby database.
- For better management and sharing of the underlying SGA/memory resources, Oracle recommends that all the Autonomous AI Databases configured for Database In-Memory be in the same Autonomous Container Database.

Refer to [Enable or Disable Database In-Memory or Create an Autonomous AI Database on Dedicated Exadata Infrastructure](#) for detailed instructions.

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](#).

Oracle LogMiner

LogMiner, which is part of Oracle Database, enables you to query online and archived redo log files through a SQL interface. Redo log files contain information about the history of activity on a database.

All changes made to user data or the database dictionary are recorded in the Oracle redo log files to perform database recovery operations. Because LogMiner provides a well-defined, easy-to-use, and comprehensive relational interface to redo log files, it can be used as a powerful data auditing tool and sophisticated data analysis tool. For key capabilities of LogMiner, refer to *LogMiner Benefits* in Oracle Database 19c Utilities or [Oracle AI Database 26ai Utilities](#).

Autonomous AI Database on Dedicated Exadata Infrastructure supports LogMiner, with the following limitations:

- The `DBMS_LOGMNR.ADD_LOGFILE` and `DBS_LOGMNR.REMOVE_LOGFILE` procedures can not be used from an Autonomous AI Database.
- The `dbms_logmnr.start_logmnr` procedure can only be invoked with SCN or time range parameters. Querying the `V$LOGMNR_CONTENTS` view subsequently returns the data from the specified SCN or time range for the connected Autonomous AI Database.

- Only one LogMiner session is allowed per Autonomous AI Database.
- By default, Autonomous AI Database archived log files are kept for *up to*:
 - 3 days for databases without GoldenGate capture and Autonomous Data Guard setup.
 - 7 days for databases with Autonomous Data Guard that do not have GoldenGate capture enabled.
 - 15 days for databases with GoldenGate capture enabled.

Therefore, LogMiner can only access up to 3, 7, or 15 days of archived log files as per the above list.

The log files retention period depends on the database transaction rate and is not always guaranteed up to the number of days specified above. There can be instances when you attempt to mine unavailable log files. In such cases, LogMiner reports ORA-1285: "error reading file".

Note

You must have the **GGADMIN** role to run LogMiner. See Unlock GGADMIN user for more details.

Oracle Text

The following describes support for Oracle Text features in Autonomous AI Database on Dedicated Exadata Infrastructure. To ensure the security and the performance of Autonomous AI Database, some Oracle Text features are restricted.

Oracle Text Feature	Supported in Autonomous AI Database	More Information
All logging, and APIs which perform logging such as <code>ctx_report.query_log_summary</code>	Not Supported	See <code>QUERY_LOG_SUMMARY</code> in Oracle Database 19c or Oracle Database 26ai
File and URL datastore	Not Supported	See <i>Datastore Type</i> in Oracle Database 19c or Oracle Database 26ai
CREATE INDEX with BIG_IO option	Not supported by default. ¹	See <i>Improved Response Time Using the BIG_IO Option of CONTEXT Index</i> in Oracle Database 19c or Oracle Database 26ai
OPTIMIZE_INDEX in rebuild mode	Not supported by default.(see Footnote 1).	<code>OPTIMIZE_INDEX</code> in Oracle Database 19c or Oracle Database 26ai

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 19c Application Developer's Guide* or [Oracle Text 26ai Application Developer's Guide](#).

Oracle XML DB

The following describes support for Oracle XML DB features in Autonomous AI Database on Dedicated Exadata Infrastructure. To ensure the security and the performance of Autonomous AI Database, some Oracle XML DB are restricted.

Note

If you migrate tables containing XMLType columns to Autonomous AI Database on Dedicated Exadata Infrastructure 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.

Area	XML DB Feature	Supported in Autonomous AI Database	More Information
Repository	XML DB Protocol	No	See <i>Repository Access Using Protocols</i> in Oracle Database 19c or Oracle Database 26ai
Repository	XML DB Resources	No	See <i>Oracle XML DB Repository Resources</i> in Oracle Database 19c or Oracle Database 26ai
Repository	XML DB ACLs	No	See <i>Repository Access Control</i> in Oracle Database 19c or Oracle Database 26ai
Storage	XML Schema Registration	No	See <i>XML Schema Registration</i> with Oracle XML DB 19c or Oracle XML DB 26ai
Storage	CLOB	No	Deprecated
Storage	Object Relational	No	See <i>XML Schema and Object-Relational XMLType</i> in Oracle Database 19c or Oracle Database 26c
Storage	Binary XML	Yes (Non schema-based only)	See <i>XMLType Storage Models</i> in Oracle Database 19c or Oracle Database 26ai
Index	Structured XML Index	Yes	See <i>XMLIndex Structured Component</i> in Oracle Database 19c or Oracle Database 26ai
Index	XQuery Full Text Index	Yes	See <i>Indexing XML Data for Full-Text Queries</i> in Oracle Database 19c or Oracle Database 26ai
Index	Unstructured XMLIndex	No	See <i>XMLIndex Unstructured Component</i> in Oracle database 19c or Oracle Database 26ai
Packages	XML DOM package	Yes	See <i>PL/SQL DOM API for XMLType (DBMS_XMLDOM)</i> in Oracle Database 19c or Oracle Database 26ai

Area	XML DB Feature	Supported in Autonomous AI Database	More Information
Packages	XML Parser Package	Yes	See <i>PL/SQL Parser API for XMLType (DBMS_XMLPARSER)</i> in Oracle Database 19c or Oracle Database 26ai
Packages	XSL Processor (DBMS_XSLPROCESSOR)	Yes	See <i>PL/SQL XSLT Processor for XMLType (DBMS_XSLPROCESSOR)</i> in Oracle Database 19c or Oracle Database 26ai

For details on Oracle XML DB, see *Oracle XML DB 19c Developer's Guide* or [Oracle XML DB 26ai Developer's Guide](#).

Oracle Flashback and Restore Points

The following describes support for Oracle Flashback and Restore Points in Autonomous AI Database on Dedicated Exadata Infrastructure. To ensure the security and the performance of Autonomous AI Database, some Oracle Flashback and Restore Point features are restricted.

To restore and recover your database to a point in time, see *Restore and Recover*.

Flashback and Restore Points Feature	Supported in Autonomous AI Database on Dedicated Exadata Infrastructure
DBMS_FLASHBACK	Yes, except for the DBMS_FLASHBACK.TRANSACTION_BACKOUT procedure.
Flashback Time Travel	Yes, except for specifying a default Flashback Data Archive using the SET DEFAULT clause with the CREATE FLASHBACK ARCHIVE and ALTER FLASHBACK ARCHIVE statements.
Flashback Drop	Yes
Flashback Query	Yes
Flashback Table	Yes
Flashback Transaction	Yes
Flashback Transaction Query	Yes
Flashback Version Query	Yes
Restore Point	No
Guaranteed Restore Point	No

For information on using Flashback features, see *About Oracle Database 19c Flashback Technology* or [About Oracle Database 26ai Flashback Technology](#).

Database Replay

You can use Database Replay to capture a workload on the production system and replay it on a test system with the original workload's exact timing, concurrency, and transaction characteristics. This enables you to test the effects of a system change without affecting the production system.

Capturing the production workload eliminates the need to develop simulation workloads or scripts, resulting in significant cost reduction and time savings. Realistic testing of complex applications that took months using load simulation tools can be completed within days with

the help of Database Replay. This enables you to rapidly test changes and adopt new technologies with higher confidence and at lower risk. For more information about Database Replay, see *Introduction to Database Replay* in Oracle 19c Testing Guide or [Oracle 26ai Testing Guide](#).

Using the Database Replay component of Real Application Testing (RAT) in the Oracle Autonomous AI Database on Dedicated Exadata Infrastructure is the same as its implementation in the Oracle Database. See *Database Replay* in Oracle Database 19c or [Oracle Database 26ai](#) for the process involved in using Database Replay with your database.

There are certain limitations and recommendations to use Database Replay with Autonomous AI Database on Dedicated Exadata Infrastructure:

- Oracle recommends that you use a Network File System (NFS) or File Storage System (FSS) that is attached to your Autonomous AI Database to store the capture and replay files. Refer to *Attach Network File System* and *Attach External File System* for guidance.
- For Database Replay, the replay client (wrc) can be the same version as the replay database, or a version lower to the replay database. However, the minimum version supported for the replay client is **19.24**.
- Autonomous AI Database does not support `dbms_workload_capture.import_awr` and `dbms_workload_replay.import_awr`. That is, you can not import Automatic Workload Repository (AWR) data with Database Replay. Hence, it is recommended that you generate the AWR comparison reports on a non-autonomous database, preferably **Oracle Database 26c instance** for better compatibility.

Database PL/SQL Packages

The following describes notes to consider before, during, and after using PL/SQL packages in Autonomous AI Database on Dedicated Exadata Infrastructure.

Lists Oracle Database PL/SQL packages with restrictions and notes in Autonomous AI Database on Dedicated Exadata Infrastructure.

Removed PL/SQL Packages

- DBMS_DEBUG_JDWP
- DBMS_DEBUG_JDWP_CUSTOM
- UTL_TCP

Note

While the UTL_TCP package is not supported on Autonomous AI Database on Dedicated Exadata Infrastructure, the UTL_TCP.CRLF constant can be used in specific instances with other packages. See *SMTP Send Email Sample Code* for an example of using UTL_TCP.CRLF.

PL/SQL Packages Notes

- UTL_HTTP Notes
 - Connections through IP addresses are allowed.
 - Egress ports 80 and 443 are enabled by default.
 - HTTP, HTTPS, and HTTP_PROXY connections are allowed.
 - The APEX_WEB_SERVICE PL/SQL package is layered on UTL_HTTP.

- Before invoking the UTL_HTTP subprograms, you must first set the wallet location using the SET_WALLET subprogram. As this is a session level setting, you must set it before each session. For more information, see *UTL_HTTP* in [Oracle Database 19c PL/SQL Packages and Types Reference](#) or [Oracle Database 26ai PL/SQL Packages and Types Reference](#).

```
BEGIN
  UTL_HTTP.SET_WALLET(path => 'file:/var/opt/oracle/dbaas_acfs/grid/
tcps_wallets');
END;
/
```

- UTL_SMTP Notes
 - Connections through IP addresses are allowed.
 - Egress ports 25, 465, 587, and 2525 are enabled by default.
 - Both simple and secure (SSL) SMTP endpoints are accessible.
 - Oracle Cloud Infrastructure (OCI) Email Delivery Service and third-party SMTP servers are both supported as email providers. See [Oracle Cloud Infrastructure - Networking and Connectivity](#) for details on establishing connectivity between the third-party server and your Autonomous AI Database tenant.
 - However, unlike Oracle Enterprise Edition on-premises database, you cannot set up localhost as the SMTP server. The `sendmail()` service is also disabled on the platform.
 - The APEX_MAIL PL/SQL package is layered on UTL_SMTP.
- DBMS_AQADM Notes
 - DBMS_AQADM.MOVE_QUEUE_TABLE can only be used in ONLINE mode to move Advanced Queuing (AQ) tables from one tablespace to another or within the same tablespace. See [My Oracle Support \(MOS\) Note: Doc ID 2867445.1](#) for guidance on moving AQ tables across tablespaces using DBMS_AQADM.MOVE_QUEUE_TABLE on 19c and above.
 - DBMS_AQADM.MOVE_QUEUE_TABLE can not be used to move AQ tables even in ONLINE mode when table or schema names are case-sensitive and include special identifiers.

Oracle Database Features That Are Not Supported in Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database on Dedicated Exadata Infrastructure supports most of the features present in Oracle Database Enterprise Edition. However, to ensure the security and the performance of Autonomous AI Database, the following Oracle Database features, options, and packs are not supported in Autonomous AI Database.

- Application containers
- Clusters (groups of tables)
- Common users
- Custom locale objects, including: language, territory, character set, and collation (linguistic sort) are not supported. Custom locale definitions created using Oracle Locale Builder cannot be deployed on Autonomous AI Database. See *Customizing Locale Data* in Oracle Database 19c or [Oracle Database 26ai](#) for more information.
- DBMS_DEBUG_JDWP and DBMS_DEBUG_JDWP_CUSTOM PL/SQL packages

- UTL_TCP PL/SQL package

Note

While the UTL_TCP package is not supported on Autonomous AI Database on Dedicated Exadata Infrastructure, the UTL_TCP.CRLF constant can be used in specific instances with other packages. See SMTP Send Email Sample Code for an example of using UTL_TCP.CRLF.

- Dictionary-managed tablespaces
- Industry Data Models
- 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 SQL Access to Kafka
- Oracle R Enterprise
- Oracle Workspace Manager
- Root container (CDB\$ROOT) access
- Transportable tablespaces
- True cache
- Uniform extent allocation
- XStream

Character Set Selection for Autonomous AI Database

Oracle recommends choosing Unicode for its universality and compatibility with contemporary and future technologies and language requirements. Autonomous AI Database on dedicated Exadata infrastructure currently uses AL32UTF8 as the database character set and AL16UTF16 as the national character set, as recommended for Oracle Multitenant.

However, for on-premises customers, with existing applications (and databases) using other character sets, migrating to Unicode character set can be a convoluted process with complex data analysis to avoid data truncation and corruption due to replacement characters. For enterprise customers using Peoplesoft, for instance, the pre-requisite of converting their data to AL32UTF8 character set as part of their migration to an Autonomous AI Database is quite complex. See *Character Set Migration* in Oracle Database 19c or [Oracle Database 26ai](#) and [Support Note 788156.1](#) for more details regarding the complexity of the exercise.

Autonomous AI Database lets you choose a character set of your choice from a list of supported character sets while provisioning an Autonomous AI Database. The list of supported character sets currently includes all database character sets supported on ASCII-based platforms. You can get the complete list of supported character sets using *ListAutonomousDatabaseCharacterSets* API. The supported national character sets supported are AL16UTF16 and UTF8.

Even though you are allowed to choose any combination of database and national character sets for an Autonomous AI Database, the Autonomous Container Database continues to use AL32UTF8 as the database character set and AL16UTF16 as the national character set. Oracle supports different database and national character sets inside an Autonomous Container Database for each database.

For step-by-step instructions on selecting a character set while creating an Autonomous AI Database, see step 12 in Create a Dedicated Autonomous AI Database.

Use Oracle Graph with Autonomous AI Database on Dedicated Exadata Infrastructure

Oracle Graph enables you to create graphs from data in an Autonomous AI Database on Dedicated Exadata Infrastructure. Graphs let you analyze your data based on connections and relationships between data entities.

As an Analyst or a Developer, you can use:

- Graph Algorithms and Graph Pattern Queries for ranking, clustering, and path analysis in a graph model of your data.
- Graph Features to detect anomalous patterns, identify communities, and find new connections in your data.

You can use graphs in your applications for fraud detection in banking, improved traceability in smart manufacturing, building linked data applications, and more. All these can be achieved along with enterprise-grade security, ease of data ingestion, and support for a wide range of workloads.

An Autonomous AI Database on Dedicated Exadata Infrastructure deployment includes all the graph capabilities from Oracle Database.

Build Applications Using Graph APIs

If you are developing an application using graphs, you can also access Property Graph APIs and RDF Graph APIs with Autonomous AI Database without Graph Studio.

- **Property Graph:** Create and work with Property Graphs with Python and Java developer APIs and the Property Graph Query Language (PGQL). You can use developer APIs to create a graph, run graph queries, and execute graph algorithms.

You can use these APIs by deploying [Oracle Graph Server and Client](#) from OCI Marketplace.

See [Oracle Database Graph](#) for more information.

- **RDF Graph:** You can manage, store, query, and perform inferencing on RDF graphs in Autonomous AI Database. Some of the key features are:
 - RDF Graph Server and Query UI: You can use a REST API, a SPARQL endpoint, a web UI to run SPARQL queries, and developer APIs for advanced RDF graph data management operations.
 - SPARQL in SQL API for integrating SPARQL and SQL queries.
 - SPARQL Editor for SQL Developer Desktop.
 - Adapter for Apache Jena.
 - Adapter for Eclipse RDF4J.

See *Graph Developer's Guide for RDF Graph* in Oracle Database 19c or [Oracle Database 26ai](#) for more details.

Use Oracle Spatial in Autonomous AI Database

Spatial features are included in Autonomous AI Database on Dedicated Exadata Infrastructure, allowing developers and analysts to get started easily with location intelligence analytics and mapping services.

About Oracle Spatial

Autonomous AI Database includes Oracle Spatial. Oracle Spatial allows developers and analysts to use spatial analysis in every application, from basic spatial search and analysis to advanced geospatial applications and Geographic Information Systems (GIS).

With the spatial features in Autonomous AI Database, organizations can:

- Manage different types of geospatial data
- Perform hundreds of spatial analytic operations, and
- Use interactive map visualization tools

The spatial features of Autonomous AI Database enable storage, retrieval, update, query, and processing of spatial data. They consist of the following:

- A schema (MDSYS) that prescribes the storage, syntax, and semantics of supported geometric data types
- A spatial indexing mechanism
- Operators, functions, and procedures for performing area-of-interest queries, spatial join queries, and other spatial analysis operations
- Utility functions and procedures for validating, loading, extracting, and working with spatial data
- GeoRaster, a feature that lets you store, index, query, analyze, and deliver GeoRaster data, that is, raster image and gridded data and its associated metadata

See *Spatial Concepts* in Oracle Database 19c or [Oracle Database 26ai](#) for more information.

Oracle Cloud Marketplace Images for Spatial

Autonomous AI Database on Dedicated Exadata Infrastructure includes Oracle Spatial with the following components.

- Geocoding, routing, and map visualization are supported through PL/SQL APIs that access Oracle Maps Cloud Service subject to the following:
 - [Oracle Maps Cloud Service Enterprise Hosting and Delivery Policies](#)
 - [End-User Terms](#)
 - [Supplier Notices](#)

See [Oracle Maps Cloud](#) for more information.

The following Oracle Spatial components require that you deploy Spatial [Oracle Cloud Marketplace](#) images in your tenancy with the Autonomous AI Database.

- [Oracle Spatial Studio](#)
- [Oracle Spatial Map Visualization](#)

- [Network Data Model](#)
- [Open Geospatial Consortium Web Services \(WFS, WCS, CSW\)](#)

Use Advanced Queuing with Autonomous AI Database

Autonomous AI Database supports Transactional Event Queues (TEQ) to implement database-integrated message queuing functionality.

Oracle Transactional Event Queues (TEQ)

Modern messaging systems support point-to-point messaging and publish/subscribe to exchanging messages and events between applications. Applications built as microservices need messaging, as do workflow systems. Event meshes are getting more prevalent to transport and route application events between applications. Like Apache Kafka's Topics/Partitions, Transactional Event Queues use Queue Tables/Event Streams (AQ Sharded Queues use Queue Tables/Shards) as the partitioning abstraction.

Transactional Event Queues (TEQ) is the partitioned and in-memory implementation of Advanced Queuing, introduced in Database 12.2.0. 1 (2015). TEQ queues support the JMS spec and can be accessed in Java, Python, Node.js, PL/SQL, C/C++, .NET, and Go.

Transactional Event Queues (TEQ) store user messages in abstract storage units called queues. When microservices communicate with each other, or applications are part of a workflow, producer applications enqueue messages, and consumer applications dequeue messages.

Oracle Transactional Event Queues (TEQ) provides *database-integrated* message queuing functionality. This functionality leverages the functions of Oracle Database so that messages can be stored persistently, propagated between queues on different computers and databases, and transmitted using Oracle Net Services and HTTP(S). For more information, see *Oracle Database Advanced Queuing* in Oracle Database 19c Advanced Queuing User's Guide or [Oracle Database 26ai Advanced Queuing User's Guide](#).

Implementing Message Queuing in Autonomous AI Database

Oracle Transactional Event Queues (TEQ) provides the following interfaces to implement message queuing:

- **Administrative Interface:** A collection of PL/SQL packages with subprograms that lets you manage queues, queue tables, transformations, and other administrative tasks such as granting and revoking privileges to TEQ users. You must connect as a user with administrative privileges to access these subprograms. To learn how to create an administrative user, see *List of Examples* in Oracle Database 19c or [Oracle Database 26ai](#).
- **Programmatic Interface:** Oracle Database Advanced Queuing offers programming interfaces for various programming languages such as PL/SQL, C, Java (JMS), and AQ XML (servlet) to implement your Advanced Queuing (AQ) application environment. For a list of AQ programmatic interfaces, functions supported in each interface, and syntax references, see *Programmatic Interfaces for Accessing Oracle Database Advanced Queuing* in Oracle Database 19c or [Oracle Database 26ai](#).

Use JSON Documents with Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database on Dedicated Exadata Infrastructure supports JavaScript Object Notation (JSON) data natively with relational database features, including transactions,

indexing, declarative querying, and views. In Autonomous AI Databases, JSON documents can coexist with relational data.

Unlike relational data, JSON data can be stored in the database, indexed, and queried without any need for a schema that defines the data. In general, you will perform the following tasks when working with JSON data in Autonomous AI Database:

1. Create a JSON column with an `is json format json check` constraint
2. Insert JSON data into the column, and
3. Query the JSON data

See *Work with JSON Documents in Autonomous AI Database* in Oracle Database 19c JSON Developer's Guide or [Oracle Database 26ai JSON Developer's Guide](#) for detailed instructions with code examples.

Note

If you use SODA to access collections in Oracle Database 19c, Oracle recommends that you use the instant client for Oracle Database 21c or later, in order to smooth migration to the use of JSON data type when your database is upgraded to release 21 or greater.

SQL and PL/SQL APIs

When you use an Autonomous AI Database to store JSON data, you can take advantage of all the features available in your Autonomous AI Database. You can combine your JSON data with non-JSON data. You can access JSON data stored in the database the same way you access other database data, including using Oracle Call Interface (OCI), Microsoft .NET Framework, and Java Database Connectivity (JDBC). See *JSON in Oracle Database 19c* or [Oracle Database 26ai](#) to learn using SQL and PL/SQL APIs to work with JSON documents.

Oracle API for MongoDB Support

Oracle Database API for MongoDB translates the MongoDB wire protocol into SQL statements that are executed by Oracle Database. It lets developers who have MongoDB skill sets write JSON document-store applications for Oracle Database that use drivers and tools that understand the MongoDB protocol. See *Overview of Oracle Database API for MongoDB* in *Oracle Database API for MongoDB* for a detailed overview of this API.

See *Use MongoDB API with Oracle Autonomous AI Database* for instructions on enabling and using the MongoDB API with Autonomous AI Database.

Simple Oracle Document Access (SODA)

Oracle also provides a family of Simple Oracle Document Access (SODA) APIs for access to JSON data stored in the database. SODA is designed for schemaless application development without knowledge of relational database features or languages such as SQL and PL/SQL. It lets you create and store collections of documents in Oracle Database, retrieve them, and query them, without needing to know how the documents are stored in the database. For more information about SODA see *Oracle as a Document Store* and *Overview of SODA*.

There are several implementations of SODA:

- **SODA for REST:** Representational state transfer (REST) requests perform collection and document operations, using any language capable of making HTTP calls.

- **SODA for Java:** Java classes and interfaces represent databases, collections, and documents.
- **SODA for PL/SQL:** PL/SQL object types represent collections and documents.
- **SODA for C:** Oracle Call Interface (OCI) handles represent collections and documents.
- **SODA for Node.js:** NoSQL style SODA APIs are used to insert, query, and retrieve SODA documents. Objects such as SodaDatabase, SodaCollection, SodaDocument, SodaDocumentCursor, and SodaOperation represent the databases, collections, documents, document cursor, and read-write operations respectively.
- **SODA for Python:** NoSQL-style cx_Oracle methods are used to insert, query, and retrieve SODA documents. Objects such as SodaDatabase, SodaCollection, SodaDocument, SodaDocumentCursor, and SodaOperation represent the databases, collections, documents, document cursor, and read-write operations respectively.

You can download the SODA drivers from the **Details** page of an Autonomous AI Database.

- Go to the **Details** page of the Autonomous AI Database. See View Details of a Dedicated Autonomous AI Database.
- Open the **Tools** tab.
- Locate the **SODA Drivers** tile, and click **Download SODA Drivers**.
- Download the SODA drivers of your choice from the displayed list of drivers.

SODA Collection Metadata

A SODA collection is a set of documents that is backed by an Autonomous AI Database table or view. By default, creating a SODA document collection creates the following in Oracle Database:

- Persistent default collection *metadata*.
- A *table* for storing the collection in the database schema to which your SODA client is connected.

You can *customize* collection metadata to obtain different behavior from those provided by default. The only metadata you can customize in Autonomous AI Database is the document-key generation method and the content media type. In particular, you cannot change the SQL data type of the column used to store JSON content (the **content column**). See SODA Collection Metadata on Autonomous AI Database to learn how to use the default and customized collection metadata.

Limitations on the Use of SODA and JSON

Autonomous AI Database supports SODA and JSON, with the following restrictions:

- Automatic indexing is not supported for SQL and PL/SQL code that uses the SQL/JSON function `json_exists`. See *SQL/JSON Condition JSON_EXISTS* in Oracle Database 19c or [Oracle Database 26ai](#) for more information.
- Automatic indexing is not supported for SODA query-by-example (QBE).

Use MongoDB API with Autonomous AI Database on Dedicated Exadata Infrastructure

Oracle Database API for MongoDB translates the MongoDB wire protocol into SQL statements that are executed by Oracle Database. It lets developers who have MongoDB skill sets write JSON document-store applications for Oracle Database that use drivers and tools that

understand the MongoDB protocol. See *Overview of Oracle Database API for MongoDB* in *Oracle Database API for MongoDB* for a detailed overview of this API.

Enable MongoDB API

You can enable the MongoDB API for an Autonomous AI Database from the OCI console or manually using Oracle REST Data Services (ORDS).

To use the MongoDB API with an Autonomous AI Database, you must install and configure customer managed Oracle REST Data Services (ORDS) separately, and the version of ORDS must be 22.3 or later. See [Oracle API for MongoDB Support](#) in *Oracle REST Data Services Installation and Configuration Guide* for the architectural concepts of this API.

To enable the MongoDB API from the OCI console:

- On the Autonomous AI Database **Details** page, select the **Tool configuration** tab.
- Click **Edit tool configuration**.
- In the **MongoDB API** row, select in the **Enable tool** column to show **Enabled**.
- Click **Apply**.

The Lifecycle state changes to **updating** until MongoDB is enabled.

You can also enable the MongoDB API when you provision or clone an instance by selecting **Show advanced options** and selecting the **Tools** tab.

To enable MongoDB manually:

- Install and configure ORDS using the `ords install adb` command.

Refer to [Installing and Configuring Customer Managed ORDS on Autonomous AI Database](#) in *Oracle REST Data Services Installation and Configuration Guide* for more details.

- Create an ORDS-enabled user.
- Connect to ORDS using the MongoDB shell.

For a demonstration of the above steps, refer to [Getting Started](#) in *Oracle REST Data Services Installation and Configuration Guide*.

Connect MongoDB Applications

After MongoDB API is enabled, you can create a MongoDB user and connect to Autonomous AI Database with its connection string.

With the MongoDB API connection string you can use the [MongoDB Shell](#), which is a command-line utility, to connect and query your data.

You can retrieve its connection string from the Oracle Cloud Infrastructure Console. To retrieve the MongoDB API connection string:

1. On the Autonomous AI Database **Details** page, select the **Tool configuration** tab.
2. In the **MongoDB API** row, under **Access URL** click **Copy**.

If you're using MongoDB API on Autonomous AI Database on Exadata Cloud@Customer, your network/DNS team must ensure that any MongoDB connection string hostname resolves to the cluster's SCAN. Without this, MongoDB clients won't be able to connect to the database. You can either allow connections to all the Autonomous AI Databases within the cluster using Wildcard DNS record or allow for a specific Autonomous AI Database:

Test MongoDB Connections

Create a Test Autonomous AI Database User for MongoDB

You can create a test user, or a similarly authenticated user, for testing purposes.

1. Open the OCI Console for your Autonomous AI Database.
2. Click **Database Actions**.
3. From the Database Actions Launchpad, click **Administration > Database Users**.
4. Click **+ Create User** on the Database Users page.
5. Enter a **User Name** and **Password** for your test user. Click **Web Access**, and set the **Quota on tablespace DATA**.
6. Click the **Granted Roles** tab.
7. In addition to the default roles, select and add the SODA_APP role for the user.
8. Click **Create User**.

Test Connection Using the Command Line

1. Login as the test user. Follow the steps in the previous example for instructions.

```
$ mongosh --tls --tlsAllowInvalidCertificates 'mongodb://  
TESTUSER:<PASSWORD>@<database URL>.  
<OCI region>.oraclecloudapps.com:27017/admin?  
authMechanism=PLAIN&authSource=$external&ssl=true&loadBalanced=false'  
Current Mongosh Log ID: 614c9e2a01e3575c8c0b2ec7  
Connecting to:      mongodb://TESTUSER:<PASSWORD>@<database  
URL>.<OCIregion>.oraclecloudapps.com:27017/admin?  
authMechanism=PLAIN&authSource=$external&tls=true&loadBalanced=false  
Using MongoDB:      3.6.2  
Using Mongosh:       1.0.7  
For mongosh info see: https://docs.mongodb.com/mongodb-shell/admin  
> show dbs  
testuser    0 B  
>
```

Note

Use URI percent-encoding to replace any reserved characters in your connection-string URI — in particular, characters in your username and password. These are the reserved characters and their percent encodings:

!	#	\$	%	&	'	(	)	*	+
%21	%23	%24	%25	%26	%27	%28	%29	%2A	%2B
,	/	:	;	=	?	@	[	]	
%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D	

For example, if your username is RUTH and your password is @least1/2#? then your MongoDB connection string to server <server> might look like this:

```
'mongodb://RUTH:%40least1%2F2%23%3F@<server>:27017/ruth/ ...'
```

Depending on the tools or drivers you use, you might be able to provide a username and password as separate parameters, instead of as part of a URI connection string. In that case you likely won't need to encode any reserved characters they contain.

See also:

- [Percent Encoding - Reserved Characters](#)
- [Uniform Resource Identifier \(URI\): Generic Syntax](#)

2. Create a collection and insert documents into your collection.

```
testuser> show collections

testuser> db.createCollection( 'fruit' )
{ ok: 1 }
testuser> show collections
fruit
testuser> db.fruit.insertOne( {name:"orange", count:42} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca31fdab254f63e4c6b47")
}
testuser> db.fruit.insertOne( {name:"apple", count:12, color: "red"} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca340dab254f63e4c6b48")
}
testuser> db.fruit.insertOne( {name:"pear", count:5} )
{
  acknowledged: true,
  insertedId: ObjectId("614ca351dab254f63e4c6b49")
}
testuser>
```

3. Query the collection using a SQL client such as Database Actions.

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

Test Connection Using a Node.js Application

1. Download Node.js. If you have already downloaded or installed Node.js for your environment, you can skip this step.

```
$ wget https://nodejs.org/dist/latest-v14.x/node-v14.17.5-linux-x64.tar.xz
```

2. Extract the contents of the Node.js archive. If you have already downloaded or installed Node.js for your environment, you can skip this step.

```
$ tar -xf node-v14.17.5-linux-x64.tar.xz
```

3. Configure the PATH environment variable. If you have already downloaded or installed Node.js for your environment, you can skip this step.

```
$ export PATH=`pwd`/node-v14.17.5-linux-x64/bin:$PATH
```

4. Test your connection with a Javascript example.

- a. Create a new directory.

```
$ mkdir autonomous_mongodb
$ cd autonomous_mongodb
$ npm init -y
```

- b. Install mongodb dependency.

```
$ npm install mongodb
```

- c. Create a JavaScript application named connect.js.

```
const { MongoClient } = require("mongodb");
const uri =
    "mongodb://TESTUSER:<PASSWORD>@<Database URI>.<OCI
    region>.oraclecloudapps.com:27017/admin?
    authMechanism=PLAIN&authSource=$external&ssl=true&loadBalanced=false";

const client = new MongoClient(uri);

async function run() {
    try {
        await client.connect();

        const database = client.db('admin');
        const movies = database.collection('movies');

        // Insert a movie
```

```

const doc = { title: 'Back to the Future',
              year: 1985, genres: ['Adventure', 'Comedy', 'Sci-
Fi'] }
const result = await movies.insertOne(doc);

// Query for a movie that has the title 'Back to the Future' :)
const query = { title: 'Back to the Future' };
const movie = await movies.findOne(query);

console.log(movie);
} finally {
  // Ensures that the client will close when you finish/error
  await client.close();
}
}
run().catch(console.dir);

```

Note

Use URI percent-encoding to replace any reserved characters in your connection-string URI — in particular, characters in your username and password. These are the reserved characters and their percent encodings:

!	#	\$	%	&	'	(	)	*	+
%21	%23	%24	%25	%26	%27	%28	%29	%2A	%2B
,	/	:	;	=	?	@	[	]	
%2C	%2F	%3A	%3B	%3D	%3F	%40	%5B	%5D	

For example, if your username is RUTH and your password is @least1/2#? then your MongoDB connection string to server <server> might look like this:

```
'mongodb://RUTH:%40least1%2F2%23%3F@<server>:27017/ruth/ ...'
```

Depending on the tools or drivers you use, you might be able to provide a username and password as separate parameters, instead of as part of a URI connection string. In that case you likely won't need to encode any reserved characters they contain.

See also:

- [Percent Encoding - Reserved Characters](#)
- [Uniform Resource Identifier \(URI\): Generic Syntax](#)

- d. Run the example. The output should look similar to the following.

```

$ node connect
{
  _id: new ObjectId("611e3266005202371acf27c1"),
  title: 'Back to the Future',
  year: 1985,

```



```
    genres: [ 'Adventure', 'Comedy', 'Sci-Fi' ]  
  }
```

About Connecting to Autonomous AI Database on Dedicated Exadata Infrastructure

Applications and tools connect to Autonomous AI Database on Dedicated Exadata Infrastructure 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 through the network defined by the dedicated infrastructure hosting the database. Oracle cloud deployments use a VCN (virtual cloud network) and subnet. Usually, the subnet is defined as private, meaning that there is no public Internet access to databases. Exadata Cloud@Customer deployments use a VM cluster network.

Secure Connections to Autonomous AI Database

Autonomous AI Database 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.

While provisioning an Autonomous Exadata VM Cluster (AVMC) resource, you can:

- Customize the Single Client Access Name (SCAN) listener port for Transport Layer Security (TLS) and non-TLS from a range of available ports (**1024 - 8999**). You can also choose mutual TLS (mTLS) authentication by selecting the **Enable mutual TLS (mTLS) authentication** check-box.
- Choose between one-way TLS and mutual TLS (mTLS) authentication modes. This is applicable only to database TLS certificates because ORDS certificates are one-way TLS certificates.

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 AI 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.

Oracle Net Services (SQL *Net) Connection Types

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 AI Database, 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.

Oracle Call Interface (OCI) Connections and Wallets

Wallet files, along with the Database user ID and password provide access to data in your Autonomous AI Database. Store wallet files in a secure location.

As wallet files can provide access to data in your database, you must share the 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.

Note

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

See Download Client Credentials for information on downloading client credentials for Autonomous AI Database.

Oracle Net Services can find the location of the Autonomous AI Database wallet using the `WALLET_LOCATION` parameter in the `sqlnet.ora` file. When `WALLET_LOCATION` is used, Oracle Net Services automatically uses the wallet. The wallet is used transparently to the application. See Prepare for ODBC and JDBC Oracle Call Interface (OCI) Connections for information on setting `WALLET_LOCATION`.

Predefined Database Service Names for Autonomous AI Databases

Autonomous AI Database on Dedicated Exadata Infrastructure provides multiple sets of database services to use when connecting to your database. In each set, one service provides a secure TCP (TCPS) connection using the TLS protocol, and another provides a TCP connection.

A connection string is created for each of the above database connection services. You can view these connection strings from the Oracle Cloud Infrastructure (OCI) console as described in View Connection Strings for an Autonomous AI Database.

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

- **tpurgent_tls** and **tpurgent**: For high priority, time critical transaction processing operations.

- **tp_tls** and **tp**: For typical transaction processing 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.

The databases configured to use **Autonomous Data Guard** have two *additional* services with the same connection types (TCPS and TCP) to the standby database for read-only access.

In a multi-standby environment, which standby database to connect to is automatically determined for **_ro** service connections. However, if needed, you can manually edit the connection string to connect to a different standby database.

Note

In the following predefined database service names, **_ro** denotes connections for read-only access, and **_ss** indicates connections to snapshot standby databases.

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

- **tpurgent_ro_tls**, **tpurgent_ro**, **tpurgent_ss_tls**, and **tpurgent_ss**: For high priority, time critical transaction processing operations.
- **tp_ro_tls**, **tp_ro**, **tp_ss_tls**, and **tp_ss**: For typical transaction processing operations.
- **high_ro_tls**, **high_ro**, **high_ss_tls**, and **high_ss**: For high priority reporting and batch operations.
- **medium_ro_tls**, **medium_ro**, **medium_ss_tls**, and **medium_ss**: For typical reporting and batch operations.
- **low_ro_tls**, **low_ro**, **low_ss_tls**, and **low_ss**: For low priority reporting and batch operations.

Autonomous AI Transaction Processing support all the above connection services (**tpurgent**, **tp**, **high**, **medium**, and **low**). In contrast, Autonomous AI Lakehouse can *only* connect to the **low**, **medium**, and **high** connection services.

Databases provisioned with CPU overprovisioning can *only* connect to:

- The **tp** and **low** services for the Autonomous AI Transaction Processing, and
- The **low** services for Autonomous AI Lakehouse.

Key Characteristics of the Predefined Database Services

- **Concurrent Statements:** The maximum number of SQL statements that an Autonomous AI Database can be processing for a consumer group before queuing the statements.
- **Degree of Parallelism:** Parallel execution is one of the Autonomous AI Database on Dedicated Exadata Infrastructure's features 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.
 - The elapsed time is critical.

- **Resource Shares:** Autonomous AI Database on Dedicated Exadata Infrastructure uses Oracle Database Resource Manager consumer groups to provide different resource allocations to each database service pair. *Resource Shares* define the fraction of resources guaranteed to a consumer group. However, any consumer group can use more than its *Resource Shares* when no resource contention exists.
- **FAN Enabled:** Indicates if Fast Application Notification is enabled by default.
- **TAC Enabled:** Indicates if Transparent Application Continuity is enabled by default.

The following table compares the different sets of database services based on the characteristics discussed above.

Service Names	Degree of Parallelism	Resource Shares	Concurrent Statements	FAN Enabled	TAC Enabled
tpurgent_tls, tpurgent, tpurgent_ro_tls, tpurgent_ro, tpurgent_ss_tls, and tpurgent_ss	Set manually	12	300 x number of OCPUs or 75 x number of ECPUs	Yes	Yes
tp_tls, tp, tp_ro_tls, tp_ro, tp_ss_tls, and tp_ss	1	8	300 x number of OCPUs or 75 x number of ECPUs	Yes	Yes
high_tls, high, high_ro_tls, high_ro, high_ss_tls, and high_ss	1 x number of OCPUs or 0.5 x number of ECPUs	4	3	Yes	No
medium_tls, medium, medium_ro_tls, medium_ro, medium_ss_tls, and medium_ss	4	2	1.25 x number of OCPUs or 0.25125 x ECPUs	Yes	No
low_tls, low, low_ro_tls, and low_ro, low_ss_tls, and low_ss	1	1	300 x number of OCPUs or 75 x number of ECPUs	Yes	No

Where the *number of OCPUs or ECPUs* is the CPU Count shown in the Oracle Cloud console.

The number of concurrent statements supported by the **tp** and **low** services with CPU overprovisioning is calculated precisely in the same way as calculated for full OCPUs or ECPUs, that is, **300 x OCPUs** or **150 x ECPUs**. For example, for a value of 0.5 OCPUs, the **total** number of concurrent statements that can be handled by **tp and low services** is 300 X 0.5, that is, 150.

Note

After connecting to your database using a service, do not attempt to switch that connection to a different service by changing the consumer group of the connection manually. Please note that the Autonomous AI Database performs more actions to configure the connection than just setting its consumer group while connecting to your database using a service.

Connect to Autonomous AI Database Using a Client Application

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

The following steps describe the process of connecting to Autonomous AI Database 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:
 - **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 AI Database and making them available to Oracle Instant Client.
See Prepare for ODBC and JDBC Oracle Call Interface (OCI) Connections for step by step instructions.
 - **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 AI Database and making them available to Oracle Database JDBC Driver.
See Prepare for JDBC Thin Connections for step by step instructions.
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.

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 AI Database connection, for example, Oracle SQL Developer, Oracle recommends that you use that type of connection.

The screenshot shows the 'New / Select Database Connection' dialog box. The 'Name' field is 'ADB Autonomous Database'. The 'Database Type' is 'Oracle'. Under 'User Info', 'Proxy User' is selected. 'Authentication Type' is 'Default'. 'Username' is 'ADMIN' and 'Role' is 'default'. The 'Password' field is masked with dots. There is a 'Save Password' checkbox. 'Connection Type' is 'Cloud Wallet'. Under 'Details', 'Advanced' is selected. 'Configuration File' is 'C:\Temp\Wallet_DB2020.zip' and 'Service' is 'db2020_medium'. A 'Configure ODS Classic' button is present. At the bottom, there are buttons for 'Help', 'Save', 'Clear', 'Test', 'Connect', and 'Cancel'.

Connect to Autonomous AI Database Using Oracle Database Tools

Oracle Database tools such as Database Actions, SQL Developer, SQLcl and SQL*Plus can be used with the Autonomous AI Database.

The following topics provide step-by-step instructions for connecting to Autonomous AI Database using Oracle Database tools:

- **Database Actions:** Connect to Autonomous AI Database with Oracle Database Actions
- **Oracle SQL Developer:** Connect to Autonomous AI Database with Oracle SQL Developer
- **Oracle SQLcl:** Connect to Autonomous AI Database with Oracle SQLcl
- **SQL*Plus:** Connect to Autonomous AI Database with SQL*Plus

Connect with Microsoft .NET, Visual Studio, and Visual Studio Code

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure supports connectivity to the Microsoft .NET Framework, .NET Core, Visual Studio, and Visual Studio Code.

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

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

- ODP.NET Core for .NET Core applications

Oracle Developer Tools for Visual Studio provides database application design-time support in Visual Studio, including tools for managing Autonomous AI Databases in Oracle Cloud.

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

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

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

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

Set-up Instructions

Refer to [Developing .NET Applications for Oracle Autonomous AI Database](#) for instructions on how to download, install, and configure these components for use with Oracle Autonomous AI Database on Dedicated Exadata Infrastructure:

Note that these instructions show how to make TCPS (TCP with SSL) connections to your database, which requires using wallets. For databases on dedicated infrastructure, you can instead make TCP connections to the database, which does not require use of a wallet.

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

Connect to Autonomous AI Database with Scripting Languages

You can use programs in different languages, including Python, Node.js, PHP, Ruby, R, Go, and Perl to connect to Oracle Autonomous AI Database on Dedicated Exadata Infrastructure. Security is enforced using client credentials.

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

Refer to the following articles for step-by-step instructions to connect to Autonomous AI Database with different scripting languages:

- **Python:** Connect Python Applications Using mTLS
- **Node.js and other scripting languages:** Connect with Node.js and other Scripting Languages

Connect with Oracle Cloud Infrastructure FastConnect

Oracle Cloud Infrastructure FastConnect provides an easy way for you to connect your on-premises network to Autonomous AI Database using FastConnect Public Peering.

FastConnect provides higher-bandwidth options, and a more reliable and consistent networking experience compared to internet-based connections.

Use FastConnect to access 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 AI Database with Oracle Cloud Infrastructure FastConnect see FastConnect Overview.

Migrate

Attach External Storage to Autonomous AI Database

Attach Network File Storage to Autonomous AI Database on Dedicated Exadata Infrastructure

You can attach your Network File Storage (NFS) share to your Autonomous AI Database, provided the database and the file system are in the same private network. Autonomous AI Database on Dedicated Exadata Infrastructure supports NFSv4, by default.

Supporting Network File System (NFS) allows you to do the following:

- Load highly sensitive data from other databases to Autonomous AI Database on Exadata Cloud@Customer in a highly secure manner.
- Move data in and out from multiple Exadata Cloud@Customer deployments to facilitate vendor data, export dumps, and other business processes.

You must attach the NFS share to the following databases explicitly, if needed:

- *Clone database* created from cloning an Autonomous AI Database with NFS mount points.
- *Standby database* that assumes the primary role after a failover or switchover operation in an Autonomous Data Guard setup where the primary database was attached to an NFS share.

Network File System Requirements

Before attaching your Network File Storage (NFS) share to an Autonomous AI Database, ensure that:

- The customer-provided NFS is network accessible from Autonomous VM Cluster nodes.
- The NFS-mounted file system is readable and writable by the `oracle` operating system user on all the Autonomous VM cluster nodes.
- If permissions are being controlled at the user level, then the `uid:gid` of the `oracle` user for the Autonomous VM cluster is `1001:1001`.

Attach Network File System

Use `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` to attach a file system to a directory in your Autonomous AI Database.

Note

The DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM procedure can only attach a private File Storage Service when the Autonomous AI Database instance is on a private endpoint.

1. Create a directory or use an existing directory to attach the external file system in your Autonomous AI Database. You must have WRITE privilege on the directory object on your Autonomous AI Database instance to attach a file system to a directory location in the database.

For example, the following command creates the database directory named NFS_DIR and creates the file system directory nfs:

```
CREATE DIRECTORY NFS_DIR AS 'nfs';
```

See Create a Directory for more information.

2. Run DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM to attach a file system to a directory in your Autonomous AI Database. To run this procedure, you must be logged in as the ADMIN user or have EXECUTE privilege on DBMS_CLOUD_ADMIN.
 - By default, DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM uses NFSv3. However, you can choose to explicitly use the params parameter and specify the nfs_version with value 3 to specify NFSv3.

```
BEGIN
DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name      => 'NFS',
    file_system_location  => 'myhost.sub000445.myvcn.oraclevcn.com:/
results',
    directory_name        => 'NFS_DIR',
    description           => 'Source NFS for sales
data'
);END;

/
```

- To use NFSv4, include the params parameter with DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM and specify the nfs_version with value 4 to specify NFSv4.

```
BEGIN
DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name => 'NFS',
    file_system_location => 'myhost.sub000445.myvcn.oraclevcn.com:/
results',
    directory_name => 'NFS_DIR',
    description => 'Source NFS for sales data',
    params => JSON_OBJECT('nfs_version' value 4)
);
END;

/
```

The examples shown above attach the network file system specified in the file_system_name parameter to the Autonomous AI Database.

The `file_system_location` parameter specifies the location of the file system. The value you supply with `file_system_location` consists of a Fully Qualified Domain Name (FQDN) and a file path in the form: `FQDN:file_path`.

For example:

- FQDN: `myhost.sub000445.myvcn.oraclevcn.com`
- File Path: `/results`

The `directory_name` parameter specifies the directory name in the Autonomous AI Database where you want to attach the file system. This is the directory you created in Step 1, or another directory you previously created.

The `description` parameter specifies the description for the task.

The `params` parameter is a JSON value that specifies an additional attribute `nfs_version`, whose value can be either 3 or 4 (NFSv3 or NFSv4).

Notes for using `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM`:

- Oracle Cloud Infrastructure File Storage uses NFSv3 to share. See [Overview of File Storage](#) for more information.
- If you attach to non-Oracle Cloud Infrastructure File Storage systems, the procedure supports NFSv3 and NFSv4.
- If you have an attached NFS server that uses NFSv3 and the NFS version is updated to NFSv4 in the NFS server, you must run `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` and then `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` (using the `params` parameter with `nfs_version` set to 4. This attaches NFS with the matching protocol so that Autonomous AI Database can access the NFSv4 Server. Without detaching and then reattaching, the NFS server will be inaccessible and you may see an error such as: "Protocol not supported".

See `ATTACH_FILE_SYSTEM` Procedure for more information.

Query `DBA_CLOUD_FILE_SYSTEMS`

After you attach a file system you can query the `DBA_CLOUD_FILE_SYSTEMS` view to retrieve information about the attached file system.

For Example:

```
SELECT file_system_name, file_system_location,  
       directory_path  
FROM dba_cloud_file_systems  
WHERE file_system_name = 'FSS';
```

This query returns details for the FSS file system name. See [DBMS_CLOUD_FILE_SYSTEMS View](#) for more information.

With an attached file system you can read and write to files on an attached file system using any PL/SQL API that accepts a directory name. For example, you can use any of the following methods to work with an attached FSS directory:

- The `UTL_FILE` package.
- Data Pump Export and Import utilities.
- The `DBMS_CLOUD` APIs that work with directories such as `DBMS_CLOUD.LIST_FILES` and `DBMS_CLOUD.PUT_OBJECT`.

Example showing a write a file on an attached file system using UTL_FILE:

```
DECLARE
  l_file          UTL_FILE.FILE_TYPE;
  l_location      VARCHAR2(100) := 'FSS_DIR';
  l_filename      VARCHAR2(100) := 'test.csv';
BEGIN
  -- Open the file.
  l_file := UTL_FILE.FOPEN(l_location, l_filename, 'w');

  UTL_FILE.PUT(l_file, 'Scott, male, 1000');

  -- Close the file.
  UTL_FILE.FCLOSE(l_file);
END;
/
```

Example showing a read a file on an attached file system using UTL_FILE:

```
DECLARE
  l_file          UTL_FILE.FILE_TYPE;
  l_location      VARCHAR2(100) := 'FSS_DIR';
  l_filename      VARCHAR2(100) := 'test.csv';
  l_text          VARCHAR2(32767);
BEGIN
  -- Open the file.
  l_file := UTL_FILE.FOPEN(l_location, l_filename, 'r');

  UTL_FILE.GET_LINE(l_file, l_text, 32767);

  -- Close the file.
  UTL_FILE.FCLOSE(l_file);
END;
/
```

Example showing list files on an attached file system using DBMS_CLOUD.LIST_FILES:

```
SELECT object_name FROM DBMS_CLOUD.LIST_FILES('FSS_DIR');
```

See the following for more information:

- *UTL_FILE* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#).
- LIST_FILES Function

DBMS_CLOUD_FILE_SYSTEMS View

The DBA_CLOUD_FILE_SYSTEMS view lists the information about the external file system attached to a directory location in the database.

Column	Data Type	Description
FILE_SYSTEM_NAME	VARCHAR2(128)	File system name
FILE_SYSTEM_LOCATION	VARCHAR2(4000)	File system location

Column	Data Type	Description
DIRECTORY_NAME	VARCHAR2(128)	Attached directory name
DIRECTORY_PATH	VARCHAR2(4000)	Attached directory path
NFS_VERSION	NUMBER	The NFS version
DESCRIPTION	VARCHAR2(4000)	The value provided for the description parameter when you run DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM
CREATION_TIME	TIMESTAMP(6) WITH TIME ZONE	Creation timestamp
UPDATE_TIME	TIMESTAMP(6) WITH TIME ZONE	Update timestamp

Detach Network File System

Use the DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM procedure to detach a file system from a directory in your Autonomous AI Database.

Note

The DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM procedure can only detach a private File Storage Service from the databases that are on private endpoints.

You must have the WRITE privilege on the directory object to detach a file system from a directory location.

Run DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM procedure to detach a file system from a directory location in your Autonomous AI Database. To run this procedure, you must be logged in as the ADMIN user or have the EXECUTE privilege on DBMS_CLOUD_ADMIN.

```
BEGIN
  DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM (
    file_system_name => 'NFS'
  );END;

/
```

This example detaches the network file system specified in the file_system_name parameter from the Autonomous AI Database. You must provide a value for this parameter. The information about this file system is removed from the DBA_CLOUD_FILE_SYSTEMS view.

See DETACH_FILE_SYSTEM Procedure for more information.

Attach External File Storage to Autonomous AI Database on Dedicated Exadata Infrastructure

You can attach an external file storage deployed in Oracle Cloud Infrastructure or on-premises to your Autonomous AI Database, provided the database and the file system are in the same

private network. Autonomous AI Database on Dedicated Exadata Infrastructure supports NFSv4, by default.

This allows you to load data from OCI File Storage in your Virtual Cloud Network (VCN) or any other Network File System in on-premises data centers.

Supporting external file storage allows you to do the following:

- Connect to an Autonomous AI Database instance from a legacy application and use the file system to load and unload data.
- Analyze data from different sources in an Autonomous AI Database.
- Secure access to data in an Autonomous AI Database from the file systems in an on-premises data center or Private VCNs.

External File Storage Requirements

To access data in an Autonomous AI Database from the file system in:

- **On-premises Data Center:** Set up FastConnect or a Site-to-Site VPN to connect to the on-premises data center. See [FastConnect](#) and [Site-to-Site VPN](#) for more information.
- **Oracle Cloud Infrastructure**
 - **Create Virtual Cloud Network (VCN) security rules for OCI File Storage Service (FSS) access:** Before you can mount a file storage, you must configure security rules to allow traffic to the mount target's VNIC using specific protocols and ports. Security rules enable traffic for the following:
 - * Open Network Computing Remote Procedure Call (ONC RPC) rpcbind utility protocol
 - * Network File System (NFS) protocol
 - * Network File System (MOUNT) protocol
 - * Network Lock Manager (NLM) protocolSee [Configuring VCN Security Rules for File Storage](#) for more details.
 - **Create a file system in OCI FSS:** Create a file system using the OCI console. When configuring mount target details for the file system, select the VCN and the private subnet of that VCN in which your database is deployed, and provide a hostname to automatically generate a Fully Qualified Domain Name (FQDN). See [Creating File Systems](#) for more information.
 - **Obtain the FQDN for the mount target:** View the details of your file system's mount target from the OCI console as described in [To view details of a mount target](#) to get its FQDN.

Attach External File System

Use `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` to attach a file system to a directory in your Autonomous AI Database.

Note

The `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` procedure can only attach a private File Storage Service to databases that are on private endpoints.

1. Create a directory or use an existing directory to attach the external file system in your Autonomous AI Database. You must have `WRITE` privilege on the directory object on your Autonomous AI Database instance to attach a file system to a directory location in the database.

For example, the following command creates the database directory named `FSS_DIR` and creates the file system directory `fss`:

```
CREATE DIRECTORY FSS_DIR AS 'fss';
```

See [Create a Directory](#) for more information.

2. Run `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` to attach a file system to a directory in your Autonomous AI Database. To run this procedure, you must be logged in as the `ADMIN` user or have `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.
 - By default, `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` uses NFSv3. However, you can choose to explicitly use the `params` parameter and specify the `nfs_version` with value 3 to specify NFSv3.

```
BEGIN
DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name      => 'FSS',
    file_system_location  => 'myhost.sub000445.myvcn.oraclecn.com:/
results',
    directory_name        => 'FSS_DIR',
    description           => 'Source FSS for sales
data'
);END;

/
```

- To use NFSv4, include the `params` parameter with `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` and specify the `nfs_version` with value 4 to specify NFSv4.

```
BEGIN
DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name => 'FSS',
    file_system_location => 'myhost.sub000445.myvcn.oraclecn.com:/
results',
    directory_name => 'FSS_DIR',
    description => 'Source FSS for sales data',
    params => JSON_OBJECT('nfs_version' value 4)
);
END;

/
```

This example attaches the network file system specified in the `file_system_name` parameter to the Autonomous AI Database.

The `file_system_location` parameter specifies the location of the file system. The value you supply with `file_system_location` consists of a Fully Qualified Domain Name (FQDN) and a file path in the form: `FQDN:file_path`. For example:

- — FQDN: `myhost.sub000445.myvcn.oraclecn.com`
- File Path: `/results`

The `directory_name` parameter specifies the directory name in the Autonomous AI Database where you want to attach the file system. This is the directory you created in Step 1, or another directory you previously created.

The `description` parameter specifies the description for the task.

Notes for using `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM`:

- Oracle Cloud Infrastructure File Storage uses NFS Version 3 to share. See [Overview of File Storage](#) for more information.
- If you attach to non-Oracle Cloud Infrastructure File Storage systems, the procedure supports NFSv3 and NFSv4.
- If you have an attached NFS server that uses NFSv3 and the NFS version is updated to NFSv4 in the NFS server, you must run `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` and then `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` (using the `params` parameter with `nfs_version` set to 4. This attaches NFS with the matching protocol so that Autonomous AI Database can access the NFSv4 Server. Without detaching and then reattaching, the NFS server will be inaccessible and you may see an error such as: "Protocol not supported".

See `ATTACH_FILE_SYSTEM` Procedure for more information.

Query `DBA_CLOUD_FILE_SYSTEMS`

After you attach a file system you can query the `DBA_CLOUD_FILE_SYSTEMS` view to retrieve information about the attached file system.

For Example:

```
SELECT file_system_name, file_system_location,  
       directory_path  
FROM dba_cloud_file_systems  
WHERE file_system_name = 'FSS';
```

This query returns details for the FSS file system name. See [DBMS_CLOUD_FILE_SYSTEMS View](#) for more information.

With an attached file system you can read and write to files on an attached file system using any PL/SQL API that accepts a directory name. For example, you can use any of the following methods to work with an attached FSS directory:

- The `UTL_FILE` package.
- Data Pump Export and Import utilities.
- The `DBMS_CLOUD` APIs that work with directories such as `DBMS_CLOUD.LIST_FILES` and `DBMS_CLOUD.PUT_OBJECT`.

Example showing a write a file on an attached file system using `UTL_FILE`:

```
DECLARE  
  l_file          UTL_FILE.FILE_TYPE;  
  l_location      VARCHAR2(100) := 'FSS_DIR';  
  l_filename      VARCHAR2(100) := 'test.csv';  
BEGIN  
  -- Open the file.  
  l_file := UTL_FILE.FOPEN(l_location, l_filename, 'w');  
  
  UTL_FILE.PUT(l_file, 'Scott, male, 1000');
```



```

    -- Close the file.
    UTL_FILE.FCLOSE(l_file);
END;
/

```

Example showing a read a file on an attached file system using UTL_FILE:

```

DECLARE
    l_file          UTL_FILE.FILE_TYPE;
    l_location      VARCHAR2(100) := 'FSS_DIR';
    l_filename      VARCHAR2(100) := 'test.csv';
    l_text          VARCHAR2(32767);
BEGIN
    -- Open the file.
    l_file := UTL_FILE.FOPEN(l_location, l_filename, 'r');

    UTL_FILE.GET_LINE(l_file, l_text, 32767);

    -- Close the file.
    UTL_FILE.FCLOSE(l_file);
END;
/

```

Example showing list files on an attached file system using DBMS_CLOUD.LIST_FILES:

```
SELECT object_name FROM DBMS_CLOUD.LIST_FILES('FSS_DIR');
```

See the following for more information:

- *UTL_FILE* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#).
- LIST_FILES Function

DBMS_CLOUD_FILE_SYSTEMS View

The DBA_CLOUD_FILE_SYSTEMS view lists the information about the external file system attached to a directory location in the database.

Column	Data Type	Description
FILE_SYSTEM_NAME	VARCHAR2(128)	File system name
FILE_SYSTEM_LOCATION	VARCHAR2(4000)	File system location
DIRECTORY_NAME	VARCHAR2(128)	Attached directory name
DIRECTORY_PATH	VARCHAR2(4000)	Attached directory path
NFS_VERSION	NUMBER	The NFS version
DESCRIPTION	VARCHAR2(4000)	The value provided for the description parameter when you run DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM
CREATION_TIME	TIMESTAMP(6) WITH TIME ZONE	Creation timestamp

Column	Data Type	Description
UPDATE_TIME	TIMESTAMP(6) WITH TIME ZONE	Update timestamp

Detach External File System

Use the `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure to detach a file system from a directory in your Autonomous AI Database.

Note

The `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure can only detach a private File Storage Service from the databases that are on private endpoints.

You must have the `WRITE` privilege on the directory object to detach a file system from a directory location.

Run `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure to detach a file system from a directory location in your Autonomous AI Database. To run this procedure, you must be logged in as the `ADMIN` user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

```
BEGIN
  DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM (
    file_system_name      => 'FSS'
  );END;

/
```

This example detaches the network file system specified in the `file_system_name` parameter from the Autonomous AI Database. You must provide a value for this parameter. The information about this file system is removed from the `DBA_CLOUD_FILE_SYSTEMS` view.

See `DETACH_FILE_SYSTEM` Procedure for more information.

Move Data Into Autonomous AI Database

Migrate Data into Autonomous AI Database on Dedicated Exadata Infrastructure

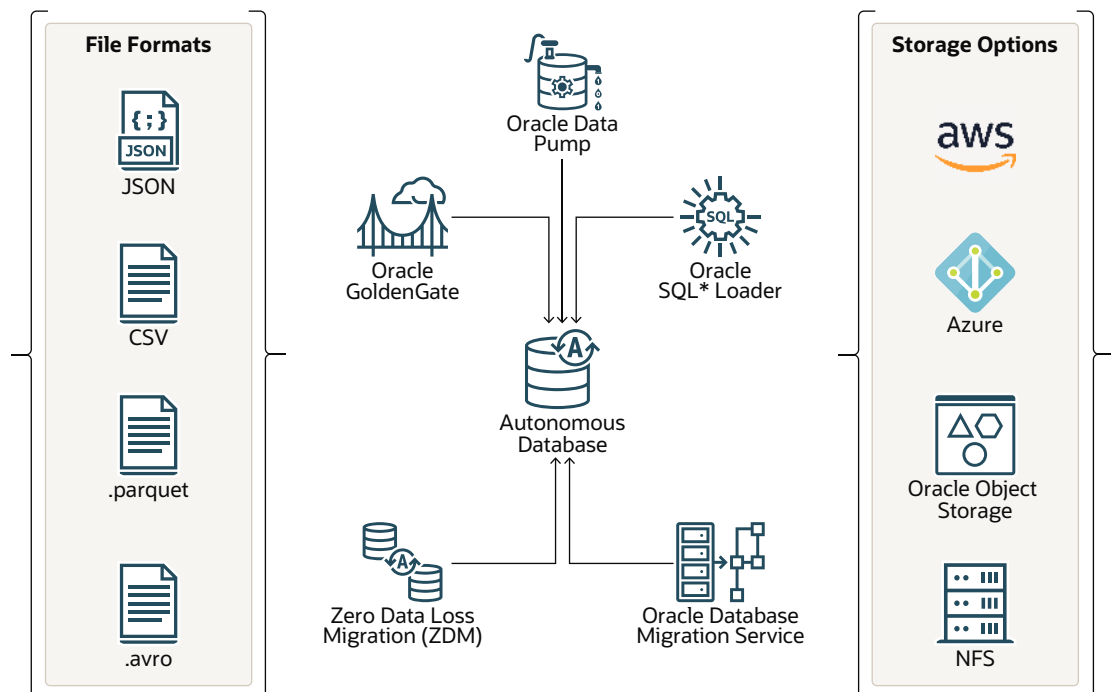
Data migration is moving data with or without its schema from one system, location, or application to another. One classic example of data migration is when you decide to adopt Autonomous AI Database for your organization and move the existing historical data from the current database to Autonomous AI Database.

Data migration is different from loading sample data or a small volume of data into your database. It usually involves large volumes of data and can be a one-time activity or periodic activity depending on the requirement. See [Data Loading Options](#) to explore the different options available for loading sample data into Autonomous AI Database.

Data Migration Overview

Oracle provides several options to migrate your data to an Autonomous AI Database on Dedicated Exadata Infrastructure. Autonomous AI Database on Dedicated Exadata Infrastructure on dedicated infrastructure lets you perform manual data export and import from:

- On-premise database, system, or data source
- SaaS applications
- Third-party data source
- Data lakes (Hadoop)
- Other Oracle Database Cloud Services



When migrating to Autonomous AI Database, you can:

- Use traditional tools such as Oracle SQL*Loader and Oracle Data Pump to migrate your source databases into Autonomous AI Database.
- Use Data Migration Service (DMS) for simple data migration of large databases that require low migration downtime.
- Use Zero Downtime Migration (ZDM) tool for secure migration to Autonomous AI Database on Public Cloud and Exadata Cloud@Customer.
- Use Oracle Object Storage to manage, load, and upload large volumes of data.
- Provide the source data in various formats such as SQL Loader text files, export-import dump files, CSV, JSON, Parquet files.
- Load data into Autonomous AI Database by attaching external Network File System (NFS) devices provided by the customer or Oracle Cloud Infrastructure File System Service (OCI FSS) using DBMS_CLOUD_ADMIN.

Note

If you notice any object compilation errors after migrating data into an Autonomous AI Database, you can recompile the database objects using UTL_RECOMP, as shown below:

```
exec sys.UTL_RECOMP.RECOMP_PARALLEL(4);
```

UTLREP.SQL cannot be used to perform a global recompile on an Autonomous AI Database.

Migration Prerequisites

As a prerequisite, it is recommended that you run the **Cloud Premigration Advisor Tool (CPAT)** to help you evaluate the compatibility of the source database with the Autonomous AI Database. CPAT identifies potential actions you may need to take before or during the migration, prioritizing their importance and suggesting resolutions. Some migration tools and services automatically run this advisor. Refer to *Cloud Premigration Advisor Tool* in Oracle Database 19c Utilities or [Oracle Database 26ai Utilities](#) for more information.

Migration Options

There are multiple options available to move your data into an Autonomous AI Database on Dedicated Infrastructure in the public cloud as well as Cloud@Customer.

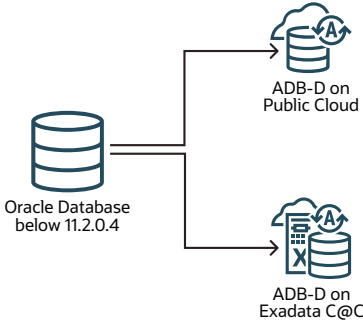
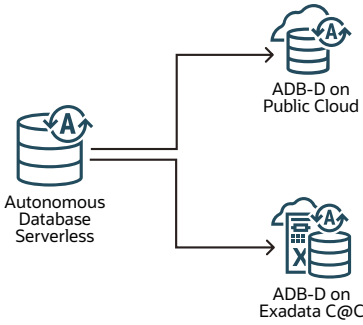
These options are listed below along with a table to pick the right option based on customer use case.

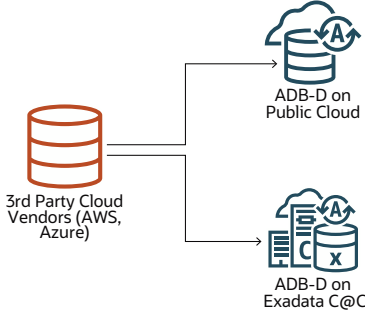
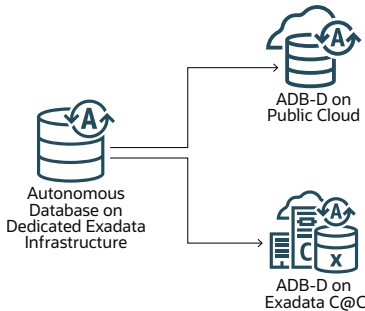
- **Oracle Database Migration Service:** A fully managed cloud service that simplifies moving established databases from on-premise, third-party, or on Oracle Cloud to Autonomous AI Database.
- **Oracle GoldenGate:** Oracle's data replication tool for one time data migration as well as data replication with change data capture.
- **Zero Downtime Migration (ZDM):** A service with a command line interface that you install and run on a host that you provision. The server where the Zero Downtime Migration software is installed is called the Zero Downtime Migration service host. You can run one or more database migration jobs from the Zero Downtime Migration service host.
- **Network File Storage (NFS):** Use your local Network File System (NFS) storage to move data in and out of Exadata Cloud@Customer deployments by attaching your Network File Storage (NFS) share to your Autonomous AI Database.
- **Oracle Data Pump:** Database utility supporting high-speed bulk data and metadata movement between Oracle databases and Autonomous AI Database.
- **Oracle SQL*Loader:** Database utility to load data from external files into Oracle Database.

Migration Use Cases

Depending on your source database, one or more of the above migration options can be used for migrating data to Autonomous AI Database on Public Cloud or Exadata Cloud@Customer deployments. Listed below are some of the most common migration use cases to help you pick the right option.

Migration Use Case	Migration Option(s)	Further Reference
Non-Oracle Database → ADB-D on Public Cloud or Exadata Cloud@Customer.	• Oracle SQL*Loader • Oracle GoldenGate • Using NFS storage to migrate data into Autonomous AI Database on Exadata Cloud@Customer • Using OCI File Storage to load data into Autonomous AI Database on Oracle Public Cloud	• Load Data Using SQL*Loader • Use Oracle GoldenGate to Replicate Data with Autonomous AI Database • Attach Network File Storage to Autonomous AI Database • Attach External File Storage to Autonomous AI Database
Oracle Database version 11.2.0.4 and above → ADB-D on Public Cloud	• Database Migration Service (DMS) • Zero Downtime Migration (ZDM) - Online or offline migration using OCI Object Storage • Oracle Data Pump • Oracle SQL*Loader • Oracle GoldenGate	• Migrate to Autonomous AI Database Using Oracle Database Migration Service • Logical Migrations with Zero Downtime Migration • Load Data Using Oracle Data Pump • Load Data Using SQL*Loader • Use Oracle GoldenGate to Replicate Data with Autonomous AI Database
Oracle Database version 11.2.0.4 and above → ADB-D on Exadata Cloud@Customer	• Zero Downtime Migration (ZDM) - Online or offline migration using OCI Object Storage • Using NFS storage to migrate data into Autonomous on Exadata Cloud@Customer • Oracle Data Pump using OCI Object Storage or Customer provided NFS Storage • Oracle GoldenGate	• Logical Migrations with Zero Downtime Migration • Use Oracle GoldenGate to Replicate Data with Autonomous AI Database • Attach Network File Storage to Autonomous AI Database

Migration Use Case	Migration Option(s)	Further Reference
Oracle Database version below 11.2.0.4 → ADB-D on Public Cloud or Exadata Cloud@Customer  Oracle Database below 11.2.0.4 ADB-D on Public Cloud ADB-D on Exadata C@C	• Oracle SQL*Loader • Oracle Data Pump • Using NFS storage to migrate data into Autonomous on Exadata Cloud@Customer • Using OCI File Storage to load data into Autonomous AI Database on Oracle Public Cloud	• Load Data Using SQL*Loader • Load Data Using Oracle Data Pump • Attach Network File Storage to Autonomous AI Database • Attach External File Storage to Autonomous AI Database
OCI Database (VMDB, BMD, ExaCS) → ADB-D on Public Cloud  OCI Database (VMDB, BMD, ExaCS) ADB-D on Public Cloud	• Database Migration Service (DMS) • Zero Downtime Migration (ZDM) - Online or offline migration using OCI Object Storage • Oracle Data Pump • Oracle SQL*Loader	• Migrate to Autonomous AI Database Using Oracle Database Migration Service • Logical Migrations with Zero Downtime Migration • Load Data Using Oracle Data Pump • Load Data Using SQL*Loader
OCI Database (VMDB, BMD, ExaCS) → ADB-D on Exadata Cloud@Customer  OCI Database (VMDB, BMD, ExaCS) ADB-D on Exadata C@C	• Zero Downtime Migration (ZDM) - Online or offline migration using OCI Object Storage • Oracle Data Pump • Oracle SQL*Loader	• Logical Migrations with Zero Downtime Migration • Load Data Using Oracle Data Pump • Load Data Using SQL*Loader
Autonomous AI Database Serverless → ADB-D on Public Cloud or Exadata Cloud@Customer  Autonomous Database Serverless ADB-D on Public Cloud ADB-D on Exadata C@C	• Oracle GoldenGate Extract configured to capture from the source Autonomous AI Database. • Oracle Data Pump	• Use Oracle GoldenGate to Replicate Data with Autonomous AI Database • Load Data Using Oracle Data Pump

Migration Use Case	Migration Option(s)	Further Reference
3rd Party Cloud Vendors (AWS, Azure) → ADB-D on Public Cloud or Exadata Cloud@Customer	• Oracle Data Pump • Zero Downtime Migration (ZDM 21.2) supports RDS and AWS	• Load Data Using Oracle Data Pump • Logical Migrations with Zero Downtime Migration
 The diagram illustrates two migration paths from 3rd Party Cloud Vendors (AWS, Azure). One path leads to ADB-D on Public Cloud, and the other leads to ADB-D on Exadata C@C. Both target databases are marked with an 'A' in a circle, indicating Autonomous AI Database.		
Autonomous AI Database on Dedicated Exadata Infrastructure → Autonomous AI Database on Dedicated Exadata Infrastructure	• Oracle GoldenGate Replicat configured to apply to the target Autonomous AI Database. • Oracle Data Pump	• Use Oracle GoldenGate to Replicate Data with Autonomous AI Database • Load Data Using Oracle Data Pump
 The diagram illustrates two migration paths from an Autonomous Database on Dedicated Exadata Infrastructure. One path leads to ADB-D on Public Cloud, and the other leads to ADB-D on Exadata C@C. Both target databases are marked with an 'A' in a circle, indicating Autonomous AI Database.		

Note

Deciding the migration tool or utility to use depends on multiple factors such as your source database, source data format, data volume and complexity. To help you identify the most optimal solution for migrating your data to Autonomous AI Database, Oracle provides an advisory utility called Oracle Cloud Migration Advisor. See www.oracle.com/goto/move for more information about this utility.

Loading Data into Autonomous AI Database on Dedicated Exadata Infrastructure

You can bulk load data into Autonomous AI Database on Dedicated Exadata Infrastructure using Oracle Database tools, and Oracle or other 3rd party data integration tools.

Data loading usually refers to loading sample data into Autonomous AI Database to explore its features or develop proof-of-concept applications.

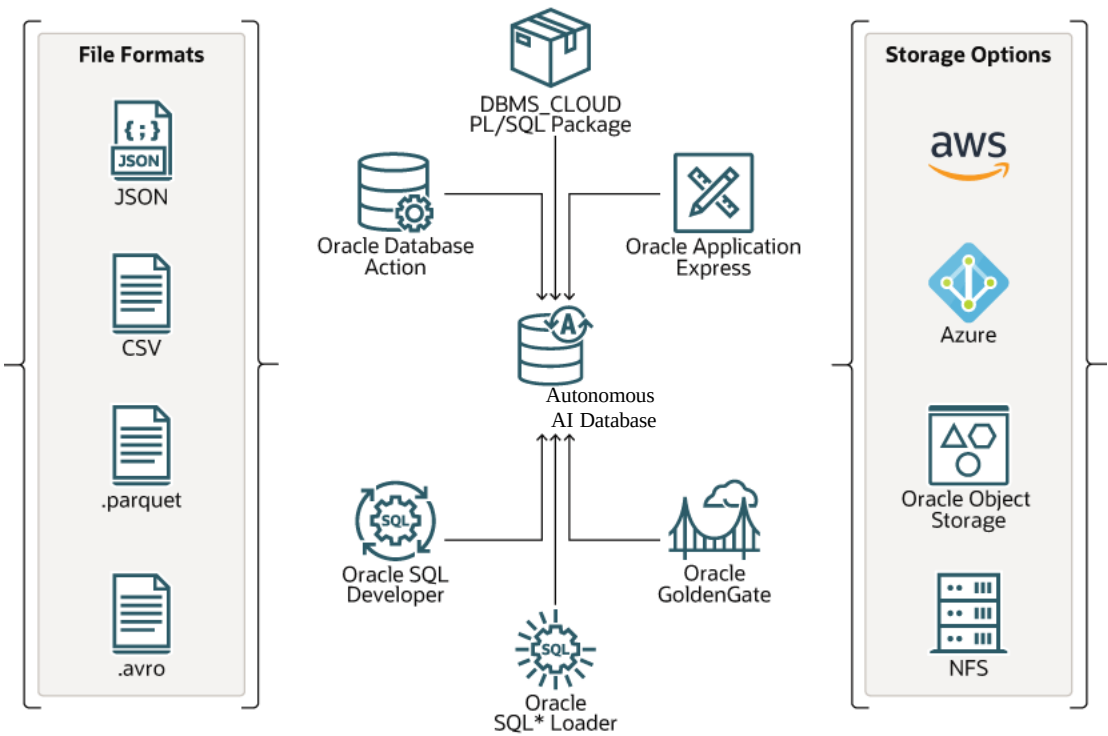
Data loading is different from data migration. Data migration usually involves large volumes of data and can be a one-time activity or routine activity depending on the requirement. See [Migrating Data into Autonomous AI Database](#) to explore the different data migration options supported by Autonomous AI Database.

Data Loading Overview

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 AI Database. Oracle provides support for loading files that are located locally in your data center, but when using this method of data loading you should factor in the transmission speeds across the Internet which may be significantly slower.

The figure below shows the most common utilities that you can use for loading data into Autonomous AI Database. You can also see the different file formats and storage options supported by the data loading options.



Loading Options

The table below lists the various data loading tools and utilities for your reference.

Tool/Utility	Description	Autonomous AI Database on Public Cloud	Autonomous AI Database on Exadata Cloud@Customer	Applicable Use Cases	For More Details
DBMS_CLOUD Package	The PL/SQL package DBMS_CLOUD provides support for loading data from files in the cloud to tables created in an Autonomous AI Database. You can load data from different file formats using the DBMS_CLOUD.COPY_DATA and the DBMS_CLOUD.COPY_TEXT procedures. External Network File System (NFS) devices provided by the customer or OCI File System Service (FSS) may be attached using the DBMS_CLOUD_ADMIN procedures.	✓	✓	Huge datasets leveraging cloud object store for fast loading.	Load Data from Files in the Cloud Attach Network File Storage to Autonomous AI Database Attach External File Storage to Autonomous AI Database
Database Actions	Web-based interface with development, data tools, administration, and monitoring features in Autonomous AI Database. Database Actions is bundled with each Autonomous AI Database instance.	✓	✓	Small test and development datasets. Load or access data from local files, from cloud storage, or from remote databases.	Loading Data from a Local File to a Table

Tool/Utility	Description	Autonomous AI Database on Public Cloud	Autonomous AI Database on Exadata Cloud@Customer	Applicable Use Cases	For More Details
Oracle SQL*Loader	Database utility to load data from external files into Oracle Database.	✓	✓	Load small amounts of data from local files in your client machine into Autonomous AI Database on Dedicated Exadata Infrastructure. One time bulk data loading from cloud storage. Move data from data lakes.	Load Data Using SQL*Loader
SQL Developer	Graphical version of SQL*Plus that gives database developers a convenient way to perform basic tasks. This tool lets you connect to schemas for MySQL and selected third-party (non-Oracle) databases, such as Microsoft SQL Server, Sybase Adaptive Server, and IBM DB2, and view metadata and data in these databases. You can then move these databases to Autonomous AI Database using PL/SQL subprograms.	✓	✓	Small test and development datasets. Adopting Oracle Autonomous AI Database.	Database Export (Unload Database Objects and Data) Deploy Objects to Cloud

Tool/Utility	Description	Autonomous AI Database on Public Cloud	Autonomous AI Database on Exadata Cloud@Customer	Applicable Use Cases	For More Details
Oracle Application Express	An easy-to-use browser-based environment to load data, manage database objects, develop REST interfaces, and build database applications for desktop and mobile devices.	✓	✓	Small test and development datasets. Load or access data from local files, from cloud storage, or from remote databases.	Loading Data Using Oracle Application Express
Oracle GoldenGate	Oracle's data replication tool for one time data migration as well as data replication with change data capture.	✓	✓	Adopting Autonomous AI Database on Dedicated Exadata Infrastructure. Scalable active-active architecture. Real-time data warehouse. Big data integration. Real-time streaming analytics.	Replicate Data Using Oracle GoldenGate

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 files in the cloud to tables created in your Autonomous AI Database on Dedicated Exadata Infrastructure.

You can load data from different file formats using the following PL/SQL procedures provided by DBMS_CLOUD:

- Text files in the cloud, using the DBMS_CLOUD.COPY_DATA procedure
- JSON files in the cloud, using the DBMS_CLOUD.COPY_TEXT procedure

Before loading the data from the files, ensure that:

- The source file is available as a local file in your client computer or uploaded to a cloud-based object store, such as Oracle Cloud Infrastructure Object Storage and is accessible to the database user attempting to load data.
- Your Cloud Object Storage credentials are stored using the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure. See [Create Credentials](#) for more information.

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

Create Credentials

Learn how to store your Cloud Object Storage credential using the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure.

Execute the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure using any database tool such as SQL*Plus, SQL Developer, or Database Actions (web based SQL Developer tool). For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_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.

Load Data from Text Files

Learn how to load data from text files in the cloud to your Autonomous AI Database using the `DBMS_CLOUD.COPY_DATA` procedure.

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_CLOUD.CREATE_CREDENTIAL` procedure. See [Create Credentials](#) for more details.

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

```
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' value ',',')
  );
END;
/
```

The parameters are:

- `table_name`: is the target table's name.
- `credential_name`: is the name of the credential created in the previous step.
- `file_uri_list`: is a comma delimited list of the source files you want to load.

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 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](#).

Load a JSON File of Delimited Documents into a Collection

Learn how to load a JSON file of delimited documents into a collection in your Autonomous AI Database using the `DBMS_CLOUD.COPY_DATA` procedure.

This example loads JSON values from a line-delimited file and uses the JSON file `myCollection.json`. Each value, each line, is loaded into a collection on your Autonomous AI Database as a single document.

Here is an example of such a file. It has three lines, with one object per line. Each of those objects gets loaded as a separate JSON document.

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

Procedure:

1. Store your Cloud Object Storage credential using the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure. See [Create Credentials](#) for more details.
2. Load data into a collection using the procedure `DBMS_CLOUD.COPY_DATA`. For example:

```

BEGIN
  DBMS_CLOUD.COPY_COLLECTION(
    collection_name => 'fruit',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/fruit_bucket/o/myCollection.json',
    format => json_object('recorddelimiter' value '''\n''')
  );
END;
/

```

The parameters are:

- `collection_name`: is the target collection'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 `myCollection.json` file in the `mybucket` bucket in the `us-phoenix-1` region. 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. The format options `characterset`, `compression`, `ignoreblanklines`, `jsonpath`, `maxdocsize`, `recorddelimiter`, `rejectlimit`, `unpackarray` are supported for loading JSON data. Any other formats specified will result in an error. For information about the format options you can specify, see [Format Parameter](#).

For more detailed information, see [COPY_COLLECTION Procedure](#).

Load an Array of JSON Documents into a Collection

Learn how to load an array of JSON documents into a collection in your Autonomous AI Database using the `DBMS_CLOUD.COPY_COLLECTION` procedure.

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

```

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

```

Procedure:

1. Store your Cloud Object Storage credential using the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure. See [Create Credentials](#) for more details.
2. Load data into a collection using the procedure `DBMS_CLOUD.COPY_DATA`. For example:

```

BEGIN
  DBMS_CLOUD.COPY_COLLECTION(
    collection_name => 'fruits',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-
string/b/json/o/fruit_array.json',
    format => '{"recorddelimiter" : "\0x'\01'", "unpackarrays" : "TRUE",
"maxdocsize" : "10240000"}'
  );
END;
/

```

In this example, you load a single JSON value which occupies the whole file. So, there is no need to specify a record delimiter. To indicate that there is no record delimiter, you can use a character that does not occur in the input file. For example, you can use value `"0x'01'"` because this character does not occur directly in JSON text.

When `unpackarrays` parameter for `format` value is set to `TRUE`, the array of documents is loaded as individual documents rather than as an entire array. The unpacking of array elements is however limited to single level. If there are nested arrays in the documents, those arrays are not unpacked.

The parameters are:

- `collection_name`: is the target collection'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 `myCollection.json` file in the `mybucket` bucket in the `us-phoenix-1` region. 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. The format options `characterset`, `compression`, `ignoreblanklines`, `jsonpath`, `maxdocsize`, `recorddelimiter`, `rejectlimit`, `unpackarray` are supported for loading JSON data. Any other formats specified will result in an error. For information about the format options you can specify, see [Format Parameter](#).

Loading `fruit_array.json` with `DBMS_CLOUD.COPY_COLLECTION` using the `format` option `unpackarrays` makes the procedure recognize array values in the source. Therefore, instead of loading the data as a single document, as it would by default, the data is loaded in the collection `fruits` with each value in the array as a single document.

For more detailed information, see [COPY_COLLECTION Procedure](#).

Copy JSON Data into an Existing Table

Use `DBMS_CLOUD.COPY_DATA` to load JSON data in the cloud into a table.

The source file in this example is a JSON data file.

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

```
SET DEFINE OFF
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'adb_user@example.com',
    password => 'password'
  );
END;
/
```

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name for all data loads.

For detailed information about the parameters, see [CREATE_CREDENTIAL Procedure](#).

2. Load JSON data into an existing table using the procedure DBMS_CLOUD.COPY_DATA.

For example:

```
CREATE TABLE WEATHER2
  (WEATHER_STATION_ID VARCHAR2(20),
   WEATHER_STATION_NAME VARCHAR2(50));
/

BEGIN
  DBMS_CLOUD.COPY_DATA(
    table_name      => 'WEATHER2',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/jsonfiles*',
    format          => JSON_OBJECT('type' value 'json', 'columnpath'
value '["$WEATHER_STATION_ID",
        "$WEATHER_STATION_NAME"]')
  );
END;
/
```

The parameters are:

- `table_name`: is the target table's name.
- `credential_name`: is the name of the credential created in the previous step.
- `file_uri_list`: is a comma delimited list of the source files you want to load. You can use wildcards in the file names in your URIs. The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character.
- `format`: for DBMS_CLOUD.COPY_DATA with JSON data, the type is json. Specify other format values to define the options to describe the format of the JSON source file. See DBMS_CLOUD Package Format Options for more information.

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

For detailed information about the parameters, see COPY_DATA Procedure.

Monitor and Troubleshoot 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, using a SELECT statement with a WHERE clause predicate on the TYPE column, shows load operations with the type COPY:

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


TABLE_NAME	OWNER_NAME	TYPE	STATUS	START_TIME	LOGFILE_TABLE	BADFILE_TABLE
UPDATE_TIME						
CHANNELS	SH	COPY	COMPLETED	04-MAR-21 07.38.30.522711000 AM GMT		04-MAR-21 07.38.30.522711000 AM GMT
		COPY\$1_LOG			COPY\$1_BAD	

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

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

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

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

Depending on the errors shown in the log and the rows shown in the specified BADFILE_TABLE table, you can correct the error by specifying the correct format options in DBMS_CLOUD.COPY_DATA.

Note

The LOGFILE_TABLE and BADFILE_TABLE tables are stored for two days for each load operation and then removed automatically.

See DELETE_ALL_OPERATIONS Procedure for information on clearing the user_load_operations table.

Textual JSON Objects That Represent Extended Scalar Values

Native binary JSON data (OSON format) extends the JSON language by adding scalar types, such as date, that correspond to SQL types and are not part of the JSON standard. Oracle Database also supports the use of textual JSON *objects* that *represent* JSON scalar values, including such nonstandard values.

When you create native binary JSON data from textual JSON data that contains such **extended objects**, they can optionally be *replaced* with corresponding (native binary) JSON scalar values.

An example of an extended object is {"\$numberDecimal":31}. It represents a JSON scalar value of the nonstandard type *decimal number*, and when interpreted as such it is replaced by a decimal number in native binary format.

For example, when you use the JSON data type constructor, JSON, if you use keyword EXTENDED then recognized extended objects in the textual input are replaced with corresponding scalar values in the native binary JSON result. If you do not include keyword EXTENDED then no such replacement occurs; the textual extended JSON objects are simply converted as-is to JSON objects in the native binary format.

In the opposite direction, when you use SQL/JSON function json_serialize to serialize binary JSON data as textual JSON data (VARCHAR2, CLOB, or BLOB), you can use keyword EXTENDED to replace (native binary) JSON scalar values with corresponding textual extended JSON objects.

Note

If the database you use is an Oracle Autonomous AI Database then you can use PL/SQL procedure `DBMS_CLOUD.copy_collection` to create a JSON document collection from a file of JSON data such as that produced by common NoSQL databases, including Oracle NoSQL Database.

If you use `ejson` as the value of the `type` parameter of the procedure, then recognized extended JSON objects in the input file are replaced with corresponding scalar values in the resulting native binary JSON collection. In the other direction, you can use function `json_serialize` with keyword `EXTENDED` to replace scalar values with extended JSON objects in the resulting textual JSON data.

These are the two main use cases for extended objects:

- *Exchange* (import/export):
 - Ingest existing JSON data (from somewhere) that contains extended objects.
 - Serialize native binary JSON data as textual JSON data with extended objects, for some use outside the database.
- *Inspection* of native binary JSON data: see what you have by looking at corresponding extended objects.

For exchange purposes, you can ingest JSON data from a file produced by common NoSQL databases, including Oracle NoSQL Database, converting extended objects to native binary JSON scalars. In the other direction, you can export native binary JSON data as textual data, replacing Oracle-specific scalar JSON values with corresponding textual extended JSON objects.

Tip

As an example of inspection, consider an object such as `{"dob" : "2000-01-02T00:00:00"}` as the result of serializing native JSON data. Is `"2000-01-02T00:00:00"` the result of serializing a native binary value of type `date`, or is the native binary value just a string? Using `json_serialize` with keyword `EXTENDED` lets you know.

The mapping of extended object fields to scalar JSON types is, in general, many-to-one: more than one kind of extended JSON object can be mapped to a given scalar value. For example, the extended JSON objects `{"$numberDecimal": "31"}` and `{"$numberLong": "31"}` are both translated as the value 31 of JSON-language scalar type `number`, and item method `type()` returns `"number"` for each of those JSON scalars.

Item method `type()` reports the JSON-language scalar type of its targeted value (as a JSON string). Some scalar values are distinguishable internally, even when they have the same scalar type. This generally allows function `json_serialize` (with keyword `EXTENDED`) to reconstruct the original extended JSON object. Such scalar values are distinguished internally either by using *different SQL types* to implement them or by *tagging them with the kind of extended JSON object* from which they were derived.

When `json_serialize` reconstructs the original extended JSON object the result is not always *textually* identical to the original, but it is always *semantically* equivalent. For example, `{"$numberDecimal": "31"}` and `{"$numberDecimal": 31}` are semantically equivalent, even

though the field values differ in type (string and number). They are translated to the same internal value, and each is tagged as being derived from a `$numberDecimal` extended object (same tag). But when serialized, the *result for both* is `{"$numberDecimal":31}`. Oracle always uses the most directly relevant type for the field value, which in this case is the JSON-language value 31, of scalar type number.

[Table 4-1](#) presents correspondences among the various types used. It maps across (1) types of extended objects used as input, (2) types reported by item method `type()`, (3) SQL types used internally, (4) standard JSON-language types used as output by function `json_serialize`, and (5) types of extended objects output by `json_serialize` when keyword `EXTENDED` is specified.

Table 4-1 Extended JSON Object Type Relations

Extended Object Type (Input)	Oracle JSON Scalar Type (Reported by <code>type()</code>)	SQL Scalar Type	Standard JSON Scalar Type (Output)	Extended Object Type (Output)
<code>\$numberDouble</code> with value a JSON number, a string representing the number, or one of these strings: "Infinity", "-Infinity", "Inf", "-Inf", "Nan" ^{1}	double	BINARY_DOUBLE	number	<code>\$numberDouble</code> with value a JSON number or one of these strings: "Inf", "-Inf", "Nan" ^{2}
<code>\$numberFloat</code> with value the same as for <code>\$numberDouble</code>	float	BINARY_FLOAT	number	<code>\$numberFloat</code> with value the same as for <code>\$numberDouble</code>
<code>\$numberDecimal</code> with value the same as for <code>\$numberDouble</code>	number	NUMBER	number	<code>\$numberDecimal</code> with value the same as for <code>\$numberDouble</code>
<code>\$numberInt</code> with value a signed 32-bit integer or a string representing the number	number	NUMBER	number	<code>\$numberInt</code> with value the same as for <code>\$numberDouble</code>
<code>\$numberLong</code> with value a JSON number or a string representing the number	number	NUMBER	number	<code>\$numberLong</code> with value the same as for <code>\$numberDouble</code>
<code>\$binary</code> with value one of these: - a string of base-64 characters - An object with fields <code>base64</code> and <code>subType</code>, whose values are a string of base-64 characters and the number 0 (arbitrary binary) or 4 (UUID), respectively When the value is a string of base-64 characters, the extended object can also have field <code>\$subType</code> with value 0 or 4, expressed as a one-byte integer (0-255) or a 2-character hexadecimal string, representing such an integer	binary	BLOB or RAW	string Conversion is equivalent to the use of SQL function <code>rawtohex</code> .	One of the following: <code>\$binary</code> with value a string of base-64 characters <code>\$rawid</code> with value a string of 32 hexadecimal characters, if input had a <code>subType</code> value of 4 (UUID)
<code>\$oid</code> with value a string of 24 hexadecimal characters	binary	RAW(12)	string Conversion is equivalent to the use of SQL function <code>rawtohex</code> .	<code>\$rawid</code> with value a string of 24 hexadecimal characters

Table 4-1 (Cont.) Extended JSON Object Type Relations

Extended Object Type (Input)	Oracle JSON Scalar Type (Reported by type())	SQL Scalar Type	Standard JSON Scalar Type (Output)	Extended Object Type (Output)
\$rawhex with value a string with an even number of hexadecimal characters	binary	RAW	string Conversion is equivalent to the use of SQL function rawtohex.	\$binary with value a string of base-64 characters, right-padded with = characters
\$rawid with value a string of 24 or 32 hexadecimal characters	binary	RAW	string Conversion is equivalent to the use of SQL function rawtohex.	\$rawid
\$oracleDate with value an ISO 8601 date string	date	DATE	string	\$oracleDate with value an ISO 8601 date string
\$oracleTimestamp with value an ISO 8601 timestamp string	timestamp	TIMESTAMP	string	\$oracleTimestamp with value an ISO 8601 timestamp string
\$oracleTimestampTZ with value an ISO 8601 timestamp string with a numeric time zone offset or with Z	timestamp with time zone	TIMESTAMP WITH TIME ZONE	string	\$oracleTimestampTZ with value an ISO 8601 timestamp string with a numeric time zone offset or with Z
\$date with value one of the following: - An integer millisecond count since January 1, 1990 - An ISO 8601 timestamp string - An object with field numberLong with value an integer millisecond count since January 1, 1990	timestamp with time zone	TIMESTAMP WITH TIME ZONE	string	\$oracleTimestampTZ with value an ISO 8601 timestamp string with a numeric time zone offset or with Z
\$intervalDaySecond with value an ISO 8601 interval string as specified for SQL function to_dsinterval	daysecondInterval	INTERVAL DAY TO SECOND	string	\$intervalDaySecond with value an ISO 8601 interval string as specified for SQL function to_dsinterval
\$intervalYearMonth with value an ISO 8601 interval string as specified for SQL function to_yminterval	yearmonthInterval	INTERVAL YEAR TO MONTH	string	\$intervalYearMonth with value an ISO 8601 interval string as specified for SQL function to_yminterval

Table 4-1 (Cont.) Extended JSON Object Type Relations

Extended Object Type (Input)	Oracle JSON Scalar Type (Reported by type())	SQL Scalar Type	Standard JSON Scalar Type (Output)	Extended Object Type (Output)
Two fields: - Field \$vector with value an array whose elements are numbers or the strings "Nan", "Inf", and "-Inf" (representing not-a-number and infinite values). - Field \$vectorElementType with string value either "float32" or "float64". These correspond to IEEE 32-bit and IEEE 64-bit numbers, respectively.	vector	VECTOR	array of numbers	Two fields: - Field \$vector with value an array whose elements are numbers or the strings "Nan", "Inf", and "-Inf" (representing not-a-number and infinite values). - Field \$vectorElementType with string value either "float32" or "float64".

¹ The string values are interpreted case-insensitively. For example, "NAN" "nan", and "nAn" are accepted and equivalent, and similarly "INF", "inFinity", and "iNf". Infinitely large ("Infinity" or "Inf") and small ("-Infinity" or "-Inf") numbers are accepted with either the full word or the abbreviation.

² On output, only these string values are used — no full-word *Infinity* or letter-case variants.

 **See Also**
[IEEE Standard for Floating-Point Arithmetic \(IEEE 754\)](#)

Use Oracle Data Pump

Oracle Data Pump offers very fast bulk data and metadata movement between Oracle databases and Autonomous AI Database on Dedicated Exadata Infrastructure.

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 AI Database.

 **Note**
For a "Try it out" alternative to reading the following topics, you can go through the *Lab 9: Migrate with Data Pump* in the Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop.

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 AI Database.

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

```
exclude=cluster, db_link  
parallel=n  
schemas=schema_name  
dumpfile=export%.dmp
```

Oracle Data Pump Export provides several export modes, Oracle recommends using the schema mode for migrating to Autonomous AI Database. You can list the schemas you want to export by using the *schemas* parameter.

For a faster migration, export your schemas into multiple Data Pump files and use parallelism. You can specify the dump file name format you want to use with the *dumpfile* parameter. Set the *parallel* parameter to at least the number of CPUs you have in your Autonomous AI Database.

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

Note

Oracle recommends that you collect the up-to-date statistics before executing the expdp command. The dbms_stats package provides multiple procedures to collect the latest statistics. Statistics are automatically added to the export dumpfile, and they help determine the size of objects and optimize parallelism during the import. For more information, see *DBMS_STATS Operational Notes* in Oracle Database 19c PL/SQL Packages and Types Reference and [Oracle Database 26ai PL/SQL Packages and Types References](#).

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

```
expdp sh/sh@orcl \  
exclude=cluster, db_link \  
parallel=16 \  
schemas=sh \  
dumpfile=export%.dmp \  
encryption_pwd_prompt=yes
```

Notes for Data Pump parameters:

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

- The `dumpfile` parameter supports the `%L` and `%l` wildcards in addition to the legacy `%U` and `%u` wildcards. For example, `dumpfile=export%L.dmp`.

Use the `%L` or `%l` wildcard for exports from Oracle Database Release 12.2 and higher. This wildcard expands the dumpfile file name into a 3-digit to 10-digit, variable-width incrementing integer, starting at 100 and ending at 2147483646.

Use the legacy `%U` or `%u` wildcard for exports from Oracle Database prior to Release 12.2. If you use this option and more than 99 dump files are needed, you must specify multiple dumpfile names, each with the `%U` or `%u` parameter.

You can use other Data Pump Export parameters, such as compression, depending on your requirements. For more information, see *Oracle Data Pump Export* in Oracle Database 19c Utilities or [Oracle Database 26ai 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 AI Database 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 AI Database 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_CLOUD.CREATE_CREDENTIAL` procedure.

Data Pump Import supports Oracle Cloud Infrastructure Auth Token based credentials and Oracle Cloud Infrastructure Signing Key based credentials.

Example 1: To create Oracle Cloud Infrastructure Auth Token credentials:

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

Example 2: To create Oracle Cloud Infrastructure Signing Key based credentials:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'DEF_CRED_NAME',
    user_ocid       => 'ocid1.user.oc1..unique_ID',
    tenancy_ocid    => 'ocid1.tenancy.oc1..unique_ID',
    private_key     => 'MIIIEogIBAAKCAQEAtUnxbmre.....wtNpESQQQ0QLGPD8NM//
JEBg=',
    fingerprint    => 'fingerprint value'
  );
END;
/
```

For more information on the credential types for Oracle Cloud Infrastructure Cloud Object Storage, see `CREATE_CREDENTIAL` Procedure.

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


```
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.

- **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://namespace-string.objectstorage.us-
ashburn-1.oci.customer-oci.com/n/namespace-string/b/bucketname/o/
export%l.dmp \
  parallel=16 \
  transform=segment_attributes:n \
  exclude=cluster, db_link
```

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%l.dmp \
  parallel=16 \
  transform=segment_attributes:n \
  exclude=cluster, db_link
```

Notes for Data Pump Import

- If during the export with `expdp` you used the `encryption_pwd_prompt=yes` parameter then use `encryption_pwd_prompt=yes` and input the same password at the `impdp` prompt that you specified during the export.
- The `dumpfile` parameter supports the `%L` and `%l` wildcards in addition to the legacy `%U` and `%u` wildcards. For example, `dumpfile=export%L.dmp`.

Use the `%L` or `%l` wildcard for exports from Oracle Database Release 12.2 and higher. This wildcard expands the dumpfile file name into a 3-digit to 10-digit, variable-width incrementing integer, starting at 100 and ending at 2147483646.

Use the legacy `%U` or `%u` wildcard for exports from Oracle Database prior to Release 12.2. If you use this option and more than 99 dump files are needed, you must specify multiple dumpfile names, each with the `%U` or `%u` parameter.
- 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](#).
- To perform a full import or to import objects that are owned by other users, you need the `DATAPUMP_CLOUD_IMP` role.

For information on disallowed objects in Autonomous AI Database, see [Limitations on the Use of SQL Commands](#).

For detailed information on the import parameters, see *Oracle Data Pump Import* in Oracle Database 19c Utilities or [Oracle Database 26ai Utilities](#).

Access Log Files for Data Pump Import

The log files for Data Pump Import operations are stored in the directory you specify with the data pump `impdp` *directory* parameter.

To access the log file you need to move the log file to your Cloud Object Storage using the procedure `DBMS_CLOUD.PUT_OBJECT`. For example, the following PL/SQL block moves the file `import.log` to your Cloud Object Storage:

```
BEGIN
  DBMS_CLOUD.PUT_OBJECT(
    credential_name => 'DEF_CRED_NAME',
    object_uri => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/import.log',
    directory_name => 'DATA_PUMP_DIR',
    file_name => 'import.log');
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

For more information, see `DBMS_CLOUD` for Objects and Files.

Use SQL*Loader

You can use Oracle SQL*Loader to load data from local files in your client machine into Autonomous AI Database on Dedicated Exadata Infrastructure.

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 AI Database. For large amounts of data Oracle recommends loading data from the Cloud Object Storage using Oracle Data Pump. 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, see *Command-Line Parameters for SQL*Loader* in Oracle Database 19c Utilities or [Oracle Database 26ai Utilities](#).

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

Autonomous AI Database 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 [Optimizer Statistics](#) or wait for the automatic statistic gathering task to kick in.

Migrate to Autonomous AI Database Using Oracle Database Migration Service

You can migrate data to an Autonomous AI Database using Oracle Cloud Infrastructure Database Migration Service that is a fully managed cloud service created to ease the move of established databases from on-premise, third-party, or on Oracle Cloud to Autonomous AI Database.

Database Migration Service provides you a high performing, self-service experience for migrating databases to Oracle Cloud Infrastructure (OCI). Database Migration runs as a managed cloud service separate from your tenancy and resources. The service leverages Oracle's Zero Downtime Migration (ZDM) engine utilizing Oracle GoldenGate replication to provide enterprise-level database migration with minimal downtime. It operates as a multitenant service in a Database Migration service tenancy and communicates with your resources using Private Endpoints (PEs). PEs are managed by Database Migration. To know about the capabilities of Database Migration Service, see [About Oracle Cloud Infrastructure Database Migration](#).

Database Migration Service supports a wide range of source databases for migrating to Autonomous AI Database. To view the complete list of supported source database versions, see [Source Database Requirements](#). As of the current release, Database Migration supports only Autonomous AI Database Serverless and Autonomous AI Database on Dedicated Exadata Infrastructure as the migration targets.

You can perform database migration in two different modes, as:

- **Online Migration:** Online migration uses Oracle GoldenGate replication for continuous synchronization from the source database to the target database. Online migration requires SQL*Net connectivity from the GoldenGate host running in the target tenancy to the source database. Oracle Data Pump and its two options are leveraged for initial instantiation. See [Online Migration](#) for details.
- **Offline Migration:** Using the offline migration method, Database Migration exports the data from the source database to the Object Store, and then imports the data from the Object Source to the target database on the Cloud using Data Pump. You must stop updates on the Source database before starting offline migration. Also, the target and source database do not need any direct SQL*Net connectivity between them, once the migration completes.

Initial Load and Replication are two important aspects of database migration using the Database Migration Service.

- **Initial Load** migrates the contents of all schemas from the source to schemas of the same name in the target database. You can elect to exclude specific objects and rename objects when you create a migration in Oracle Cloud Infrastructure Database Migration.
- **Replication** migrates all data and metadata operations in transactions committed after the initial load until you resume the migration job after the Monitor Replication Lag phase. It includes inserts, deletes, and updates of data in tables within the migrated schema. Create, alter, and drop DDL operations are not replicated.

To learn how to use Database Migration Service, refer to [Database Migration: Prerequisites and Steps Involved in Using Oracle Cloud Infrastructure Database Migration Service](#).

Use Oracle GoldenGate to Replicate Data with Autonomous AI Database

You can replicate data to Autonomous AI Database using Oracle GoldenGate and Oracle GoldenGate Cloud Service. You can capture changes from the Autonomous AI Database and

replicate to any target database or platform that Oracle GoldenGate supports, including other Autonomous AI Database on Dedicated Exadata Infrastructures.

Note: Backups must be enabled on the source Autonomous Container Database to guarantee that all source operations are properly recorded in the redo logs. If backups are not enabled, **NOLOGGING** operations may not be captured by Oracle GoldenGate, which can lead to data loss or data inconsistency on the target system.

Additionally, configure a **backup retention period** that allows archived redo logs to be restored from backup if they are removed from the database file system due to space constraints.

Oracle GoldenGate offers efficient data replication solutions in multiple scenarios, some of which are listed below:

- Data migration to or between Autonomous AI Databases
- Real-time data capture and replication to support:
 - Scalable active-active architecture
 - Real-time data warehouse
 - Big data integration
 - Real-time streaming analytics

Depending on whether your Autonomous AI Database is configured as the source database (extract) or target database (replicat) in the data replication setup, you will have to perform one or more tasks listed below:

- **Unlock the GGADMIN account:** GGADMIN is a pre-created Oracle GoldenGate database user in Autonomous AI Database. This account is used for both Extracts and Replicats in the Autonomous AI Database and must be unlocked before configuring data replication with Oracle GoldenGate. For more details, see *Establishing Oracle GoldenGate Credentials*.
- **Configure Extract to capture from an Autonomous AI Database:** This is required only if you want to use Autonomous AI Database as the source database in the data replication activity. For complete details, see *Configuring Extract to Capture from an Autonomous AI Database*.
- **Configure Replicat to apply to an Autonomous AI Database:** This is required only if you want to use Autonomous AI Database as the target database in the data replication activity. For complete details, see *Configuring Replicat to Apply to an Autonomous AI Database*.

Parallel Replicat is a variant of Replicat that applies transactions in parallel to improve performance. Autonomous AI Database supports Parallel Replicat in integrated mode, that is, you can also create (or add) and configure parallel replication in your environment. See *Using Parallel Replicat* for more information.

 Note

Parallel Replicat is not supported for overprovisioned Autonomous AI Databases, that is, databases provisioned with fractional OCPUs (less than 1 OCPU).

 Tip

For a "Try it out" alternative that demonstrates live migration to Autonomous AI Database using Oracle GoldenGate replication, run *Lab 10: Oracle GoldenGate Replication* in the Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop.

Decrypt Data While Importing from Object Storage

You can decrypt and load data from encrypted files stored in Object Storage. You can also decrypt encrypted data on Object Storage that you use in an external table.

This option is useful when migrating from an on-premises database to an Autonomous AI Database if the data in your source files is encrypted.

 Note

This option is only supported for Object Storage files less than 4 GB.

This option is applicable for the following procedures:

- `DBMS_CLOUD.COPY_DATA`
- `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`
- `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`
- `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE`

For `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` this option is only applicable to the Object Storage files.

- `DBMS_CLOUD.COPY_COLLECTION`

Decrypt and Load Data Using DBMS_CRYPTO Algorithms

Shows the steps to decrypt encrypted files from Object Storage and load the data into a table on Autonomous AI Database (the decrypt step uses `DBMS_CRYPTO` algorithms).

As a prerequisite you must have encrypted files and uploaded the files into Object Storage. This example uses a CSV file and it is assumed that the file is encrypted using `DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5` algorithm and uploaded to your Cloud Object Storage.

See [ENCRYPT Function](#) for more information on the `ENCRYPT` function.

See [DBMS_CRYPTO Operational Notes](#) for more information on generating an encryption key.

To decrypt and load data into an existing table on Autonomous AI Database from Object Storage:

1. Connect to your Autonomous AI Database instance.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OBJ_STORE_CRED',
    username        => 'user1@example.com',
    password        => 'password'
  );
END;
/
```

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

See `CREATE_CREDENTIAL` Procedure for more information.

3. Create a credential to store the key using `DBMS_CLOUD.CREATE_CREDENTIAL`. For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'ENC_CRED_NAME',
    username        => 'Any_username',
    password        => 'password'
  );
END;
/
```

As an alternative you can create credentials to store the key in a vault. For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'ENC_CRED_NAME',
    params          => JSON_OBJECT ('username' value 'Any_username',
                                   'region'    value 'Region',
                                   'secret_id' value 'Secret_id_value'));
END;
/
```

Note

The username parameter you specify in the credential that stores the key can be any string.

This creates the `ENC_CRED_NAME` credential which is a vault secret credential, where the secret (decryption/encryption key) is stored as a secret in the Oracle Cloud Infrastructure Vault.

See `CREATE_CREDENTIAL` Procedure for more information.

4. Run `DBMS_CLOUD.COPY_DATA` and specify `DBMS_CRYPTO` encryption algorithm as decryption method.

```
BEGIN
  DBMS_CLOUD.COPY_DATA (
```

```

        table_name      => 'CSV_COPY_DATA',
        credential_name => 'OBJ_STORE_CRED',
        file_uri_list   => 'https://objectstorage.us-
        phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/encrypted.csv',
        format          => json_object( 'type' value 'csv', 'encryption' value
        json_object( 'type' value DBMS_CRYPTO.ENCRYPT_AES256 +
        DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5, 'credential_name' value
        'ENC_CRED_NAME' ))
    );
END;
/

```

This decrypts the ENCRYPTED.CSV file in the Object Storage. The data is then loaded into the CSV_COPY_DATA table. The format parameter encryption option value specifies a DBMS_CRYPTO encryption algorithm to be used to decrypt data.

See [DBMS_CRYPTO Algorithms](#) for more information on encryption algorithms.

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

For detailed information about the parameters, see COPY_DATA Procedure.

For detailed information about the available format parameters, you can use with DBMS_CLOUD.COPY_DATA, see [#unique_306](#).

Decrypt and Load Data with a User Defined Function

Shows the steps to decrypt files in Object Storage and load the data into tables using a user-defined decryption function.

As a prerequisite for these steps you must have encrypted files and uploaded the files into Object Storage. This example uses a CSV file and it is assumed that the file is encrypted using DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5 algorithm and uploaded to your Cloud Object Storage.

See [ENCRYPT Function](#) for more information on the ENCRYPT function.

See [DBMS_CRYPTO Operational Notes](#) for more information on generating an encryption key.

To decrypt and load data into an existing table on Autonomous AI Database from Object Storage:

1. Connect to your Autonomous AI Database instance.
2. Store your Cloud Object Storage credential using DBMS_CLOUD.CREATE_CREDENTIAL.

For example:

```

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OBJ_STORE_CRED',
        username        => 'user1@example.com',
        password         => 'password'
    );
END;
/

```

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

See CREATE_CREDENTIAL Procedure for more information.

3. Create a user-defined function decryption callback function.

For example:

```
CREATE OR REPLACE FUNCTION decryption_func(data IN BLOB)
RETURN BLOB
IS
    l_decrypted_data BLOB;
BEGIN
    DBMS_LOB.createtemporary(l_decrypted_data, TRUE, DBMS_LOB.CALL);
    DBMS_CRYPTO.decrypt(
        dst => l_decrypted_data,
        src => data,
        typ => DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC +
        DBMS_CRYPTO.PAD_PKCS5,
        key => 'encryption key'
    );
    RETURN l_decrypted_data;
END decryption_callback;
/
```

This creates the DECRYPTION_FUNC decryption function. This function decrypts data using a stream or block cipher with a user supplied key. The user supplied key in the example is stored in Oracle Cloud Infrastructure Vault and is retrieved dynamically by making a REST call to Oracle Cloud Infrastructure Vault service.

4. Run DBMS_CLOUD.COPY_DATA and specify the format option encryption and specify the user-defined function you created to decrypt the data.

```
BEGIN
    DBMS_CLOUD.COPY_DATA (
        table_name      => 'CSV_COPY_DATA',
        credential_name => 'OBJ_STORE_CRED',
        file_uri_list   => 'https://objectstorage.us-
        phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/encrypted.csv',
        format          => json_object(
            'type' value 'csv',
            'encryption' value
            json_object('user_defined_function' value 'admin.decryption_func'))
    );
end;
/
```

This decrypts the ENCRYPTED.CSV file in the Object Storage. The data is then loaded into the CSV_COPY_DATA table. The format parameter encryption option value specifies a user-defined function name to use to decrypt data.

Note

You must have the EXECUTE privilege on the user-defined function.

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

For detailed information about the parameters, see COPY_DATA Procedure.

For detailed information about the available format parameters, you can use with DBMS_CLOUD.COPY_DATA, see DBMS_CLOUD Package Format Options.

Export Data from Autonomous AI Database

Export Data from Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database on Dedicated Exadata Infrastructure offers several options to export your data if you choose to move to another cloud service provider or transition to an on-premises database environment.

This flexibility supports your organization's evolving data management needs, ensuring that you maintain access and control over your information throughout its life cycle. Whether you aim to migrate a full database or only selected datasets, an Autonomous AI Database provides established methods and tools to facilitate data export, integration, and transfer across platforms.

Below are resources and documentation that guide you through the process of moving your data:

- [Export Data Using Oracle Data Pump: Review the options to move data from an Autonomous AI Database using Oracle Data Pump.](#)
- [Export Data to a Directory Using DBMS_CLOUD.EXPORT_DATA](#) : Access detailed instructions for exporting files to a directory using DBMS_CLOUD.EXPORT_DATA.
- [Export Data to Object Store as Text Using DBMS_CLOUD.EXPORT_DATA](#): Learn how to use DBMS_CLOUD.EXPORT_DATA to export data as text from an Autonomous AI Database to cloud Object Store.

After reviewing these resources, you can choose the method that best fits your technical requirements and migration objectives. The documentation links above offer step-by-step guidance to support you through each stage of the migration process.

Use Oracle Data Pump to Export Data from Autonomous AI Database

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

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous AI Database to other Oracle databases, as it contains enhancements and fixes for a better experience. Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform from [Oracle Instant Client Downloads](#). See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package.

To move data from an Autonomous AI Database to other Oracle databases using Oracle Data Pump, use one of the following options:

- **Option 1: Move Data with Data Pump Export to an Autonomous AI Database Directory**

Use Oracle Data Pump to export the data to a directory on your database, and then move the data from the directory to Cloud Object Store. This method can be beneficial when you

want to move data from your Autonomous AI Database to multiple target databases. Instead of exporting the data to multiple Cloud Object Stores, you can create a dump file set once and use it for multiple target databases.

Perform the following steps to move data from Autonomous AI Database by exporting it to a directory on your database:

1. Export data to a directory on Autonomous AI Database and move the dump file set from the directory to your Cloud Object Store. See [Export Data Using a Directory on Autonomous AI Database](#).
2. Download the dump files from the Cloud Object Store, import the data into the target database, and clean up the Cloud Object Store. See [Download Dump Files, Run Data Pump Import, and Clean Up Object Store](#).

- **Option 2: Move Data with Data Pump Export to Object Store**

Using this export method you use Oracle Data Pump to directly export data to your object store. This export method is supported with Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic. This avoids the overhead of creating a dump file set in a directory on your Autonomous AI Database and then moving the dump file set to the Cloud Object Store. When you plan to move data to a single target database, this method saves effort and speeds up the export process.

Perform the following steps to move data from Autonomous AI Database by exporting it directly to your object store:

1. Export data from Autonomous AI Database to a Cloud Object Store directly. See [Export Data From Autonomous AI Database to Object Store Directly](#).
2. Download the dump files from the Cloud Object Store, import the data into the target database, and clean up the Cloud Object Store. See [Download Dump Files, Run Data Pump Import, and Clean Up Object Store](#).

Export Data Using a Directory on Autonomous AI Database

To export data using a directory on Autonomous AI Database, you must first create a dump file set with the exported data on a directory in your database and then upload these files from your database directory to the Cloud Object Store.

Use Data Pump to Create a Dump File Set on Autonomous AI Database

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 50G, and the *directory* parameter set. For example, the following shows how to export a schema named SALES in an Autonomous AI Database named ATPC1 with 64 ECPU's:

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

Notes for Data Pump parameters:

- In case of ECPUs, `parallel` must be set to 0.25 times the ECPU count. Hence, in the above example with 64 ECPUs, `parallel` is set to 0.25×64 , that is, 16 with `expdp`.
- In case of OCPUs, `parallel` must be set to the same value as OCPU count. In the above example with 16 OCPUs, `parallel` will be set to 16, with `expdp`.
- 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.
- The `dumpfile` parameter supports the `%L` and `%l` wildcards in addition to the legacy `%U` and `%u` wildcards. For example, `dumpfile=export%L.dmp`.

Use the `%L` or `%l` wildcard for exports from Oracle Database Release 12.2 and higher. This wildcard expands the dumpfile file name into a 3-digit to 10-digit, variable-width incrementing integer, starting at 100 and ending at 2147483646.

For the best export performance use the HIGH database service for your export connection and set the `PARALLEL` parameter to the number of CPUs in your a database. For information on which database service name to connect to run Data Pump Export, see [Predefined Database Service Names for Autonomous AI Databases](#).

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

```
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:

OBJECT_NAME LAST_MODIFIED	BYTES	CHECKSUM	CREATED
exp01.dmp NOV-19...	12288		12-NOV-19 06.10.47.0 PM GMT 12-
exp02.dmp NOV-19...	8192		12-NOV-19 06.10.48.0 PM GMT 12-
exp03.dmp NOV-19...	1171456		12-NOV-19 06.10.48.0 PM GMT 12-
exp04.dmp NOV-19...	348160		12-NOV-19 06.10.48.0 PM GMT 12-
export.log NOV-19...	1663		12-NOV-19 06.10.50.0 PM GMT 12-

Notes

- To perform a full export or to export objects that are owned by other users, you need the DATAPUMP_CLOUD_EXP role.
- The API you use to move the dump files to your Object Storage supports file sizes up to 50GB, so make sure you do not specify sizes higher than that in the *filesize* argument.
- For more information, see *Oracle Data Pump Export* in Oracle Database 19c Utilities or [Oracle Database 26ai Utilities](#).

Move Dump File Set from Autonomous AI Database to Your Cloud Object Store

1. Connect to your Autonomous AI 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 => 'adb_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 AI 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.

Export Data From Autonomous AI Database to Object Store Directly

Shows how to use Oracle Data Pump to export data from Autonomous AI Database to Cloud Object Store directly.

1. Connect to your Autonomous AI 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 => 'adb_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. As an ADMIN user, set the credential you defined in the above step as the default credential for your Autonomous AI Database.

For example:

```
ALTER DATABASE PROPERTY SET DEFAULT_CREDENTIAL = 'DEF_CRED_NAME';
```

4. Run Data Pump Export with the *dumpfile* parameter set to the URL for an existing bucket on your Cloud Object Storage (ending with a file name or a file name with a substitution variable, such as exp%U.dmp).
 - **Oracle Data Pump version 19.9 or later:**
set the credential parameter to the name of the credential you created in Step 2. For example:

```
expdp admin/password@ADBD_high \
SCHEMAS=SOE3 \
filesize=5GB \
credential=DEF_CRED_NAME \
dumpfile=https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/
adbdpreview1/mybucket/export%L.dmp \
parallel=16 \
encryption_pwd_prompt=yes \
logfile=export.log \
directory=data_pump_dir \
EXCLUDE=statistics,index
```

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-ashburn-1* region. (*adbdpreview1* is the object storage namespace in which the bucket resides.)

- **Oracle Data Pump versions earlier than 19.9:**
start the value of the *dumpfile* parameter with the *default_credential* keyword and a colon. For example:

```
expdp admin/password@ADBD_high \
SCHEMAS=SOE3 \
filesize=5GB \
dumpfile=DEF_CRED_NAME:https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/adbdpreview1/mybucket/export%L.dmp \
parallel=16 \
encryption_pwd_prompt=yes \
logfile=export.log \
directory=data_pump_dir \
EXCLUDE=statistics,index
```

In this example, the *default_credential* keyword in the *dumpfile* parameter is required.

Notes for Oracle Data Pump parameters:

- By default, Oracle Data Pump Export compresses only metadata before writing to the dump file set. You can set the compression parameter to *ALL* to compress metadata and data before writing to the dump file set. See *COMPRESSION* in Oracle Database 19c Utilities or [Oracle Database 26ai Utilities](#) for more details.
- The *dumpfile* parameter supports the %L and %l wildcards in addition to the legacy %U and %u wildcards. For example, *dumpfile=export%L.dmp*.

Use the %L or %l wildcard for exports from Oracle Database Release 12.2 and higher. This wildcard expands the dumpfile file name into a 3-digit to 10-digit, variable-width incrementing integer, starting at 100 and ending at 2147483646.

For more information, see *Parameters Available in Data Pump Export Command-Line Mode* in Oracle Database 19c Utilities or [Oracle Database 26ai Utilities](#).

- For the best export performance use the high database service for your export connection and set the `parallel` parameter to 0.25 times the number of ECPU's or same as the number of OCPU's in your Autonomous AI Database. For information on which database service name to connect to run Data Pump Export, see *Predefined Database Service Names for Autonomous AI Databases*.
- 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 AI Database, see *Limitations on the Use of SQL Commands*.

5. Validate the results.

Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:

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

Note

Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token. For example:

```
curl -O -v -X GET -u 'user1@example.com:auth_token' \
https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/
namespace-string/bucketname/export01.dmp
```

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 another Autonomous AI Database.

If you export directly to Object Store using Oracle Data Pump, as shown in [Export Data From Autonomous AI Database to Object Store Directly](#), then the dump files on Object Store show size 0. Oracle Data Pump divides each dump file part into smaller chunks for faster uploads. The Oracle Cloud Infrastructure Object Storage console shows multiple files for each dump file part that you export. The size of the actual dump files will be displayed as zero (0) and its related file chunks as 10mb or less. For example:

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

Downloading the zero byte dump file from the Oracle Cloud Infrastructure console or using the Oracle Cloud Infrastructure CLI will not give you the full dump files. To download the full dump files from the Object Store, use a tool that supports Swift such as curl, and provide your user login and Swift auth token. For example:

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

The cURL command does not support wildcards or substitution characters in its URL. You need to use multiple cURL commands to download the dump file set from your Object Store. Alternatively, you can use a script that supports substitution characters to download all the dump files from your Object Store in a single command. See [How To: Download all files from an export to object store job in Autonomous AI Database using cURL](#) for an example.

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

Note

If you are importing the data to another Autonomous AI Database, see Load Data Using Oracle Data Pump.

In case of files exported directly to Object Store using Oracle Data Pump, as shown in [Export Data From Autonomous AI Database to Object Store Directly](#), if you import a file with the DBMS_CLOUD procedures that support the format parameter type with the value 'datapump', you only need to provide the primary file name. The procedures that support the 'datapump' format type automatically discover and download the chunks.

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, see *Parameters Available in Oracle Data Pump Import Command-Line Mode* in Oracle Database 19c Utilities or [Oracle Database 26ai Utilities](#).

Export Data to a Directory

Use DBMS_CLOUD.EXPORT_DATA to export files to a directory.

The directory to which you export files can be in the Autonomous AI Database file system or an attached external file system. See the following pages for more information:

- [Create and Manage Directories](#)
- [Attach Network File Storage to Autonomous AI Database](#)
- [Attach External File Storage to Autonomous AI Database](#)

Export Data as CSV to a Directory

Shows the steps to export table data from your Autonomous AI Database to a directory as CSV data by specifying a query.

1. Connect to your Autonomous AI Database instance.
See [Connect to Autonomous AI Database](#) for more information.

2. Create a directory.

For example:

```
CREATE DIRECTORY export_dir AS 'export_dir';
```

See [Create a Directory](#) for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the format parameter type with the value `json` to export the results as CSV files to a directory. Do not include the credential parameter when sending output to a directory.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales.csv',
    format => JSON_OBJECT('type' value 'csv'),
    query => 'SELECT * FROM sales'
  );
END;
/
```

When record delimiters include escape characters, such as `\r\n` or `\t`, enclose the record delimiters in double quotes. For example, to use the record delimiter `\r\n`, enclose the value in double quotes: `"\r\n"`.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales.csv',
    query => 'SELECT * FROM sales',
    format => JSON_OBJECT('type' value 'json', 'recorddelimiter' value
      '"\r\n"' format json));
END;
/
```

The directory name is case-sensitive when the directory name is enclosed in double quotes. For example:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => '"export_dir":sales.csv',
```

```
format => JSON_OBJECT('type' value 'csv'),  
query => 'SELECT * FROM sales'  
);  
END;  
/
```

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

For detailed information about the available format parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see `DBMS_CLOUD` Package Format Options for `EXPORT_DATA`.

The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required type parameter. The valid values are `json`, `xml`, and `csv`.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

Note

The `DBMS_CLOUD.EXPORT_DATA` procedure creates the dump file(s) that you specify in the `file_uri_list`. The procedure does not overwrite files. If a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` generates another file with a unique name. `DBMS_CLOUD.EXPORT_DATA` does not create directories.

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files.

Export Data as JSON a Directory

Shows the steps to export table data from your Autonomous AI Database to a directory as JSON data by specifying a query.

1. Connect to your Autonomous AI Database instance.
See [Connect to Autonomous AI Database](#) for more information.
2. Create a directory.

For example:

```
CREATE DIRECTORY export_dir AS 'export_dir';
```

See [Create a Directory](#) for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the format parameter type with the value `json` to export the results as JSON files to a directory. Do not include the credential parameter when sending output to a directory.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales.dmp',
    format => json_object('type' value 'json'),
    query => 'SELECT * FROM sales'
  );
END;
/
```

When record delimiters include escape characters, such as `\r\n` or `\t`, enclose the record delimiters in double quotes. For example, to use the record delimiter `\r\n`, enclose the value in double quotes: `"\r\n"`.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales.dmp',
    query => 'SELECT * FROM sales',
    format => JSON_OBJECT('type' value 'json', 'recorddelimiter' value
      '"\r\n"' format json));
END;
/
```

The directory name is case-sensitive when the directory name is enclosed in double quotes. For example:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => '"export_dir":sales.dmp',
    format => json_object('type' value 'json'),
    query => 'SELECT * FROM sales'
  );
END;
/
```

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

For detailed information about the available format parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see `DBMS_CLOUD` Package Format Options for `EXPORT_DATA`.

The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required type parameter. The valid values are `json`, `xml`, and `csv`.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

Note

The `DBMS_CLOUD.EXPORT_DATA` procedure creates the dump file(s) that you specify in the `file_uri_list`. The procedure does not overwrite files. If a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` generates another file with a unique name. `DBMS_CLOUD.EXPORT_DATA` does not create directories.

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files.

Export Data as Parquet to a Directory

Shows the steps to export table data from your Autonomous AI Database to a directory as Parquet data by specifying a query.

1. Connect to your Autonomous AI Database instance.

See [Connect to Autonomous AI Database](#) for more information.

2. Create a directory.

For example:

```
CREATE DIRECTORY export_dir AS 'export_dir';
```

See [Create a Directory](#) for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the format parameter type with the value `json` to export the results as Parquet files to a directory. Do not include the `credential` parameter when sending output to a directory.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales.parquet',
    format => JSON_OBJECT('type' value 'parquet'),
    query => 'SELECT * FROM sales'
  );
END;
```

The directory name is case-sensitive when the directory name is enclosed in double quotes. For example:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => '"export_dir":sales.parquet',
    format => JSON_OBJECT('type' value 'parquet'),
    query => 'SELECT * FROM sales'
  );
```

```
END;  
/
```

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

For detailed information about the available format parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see `DBMS_CLOUD` Package Format Options for `EXPORT_DATA`.

The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required type parameter. The valid values are `datapump`, `json`, `xml`, `csv` and `parquet` and it also optionally defines the options you can specify for the export with the `ORACLE_DATAPUMP` Access Driver.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files.
- See `DBMS_CLOUD` Package Oracle Data Type to Parquet Mapping for details on Oracle Type to Parquet Type mapping.

The following types are not supported or have limitations on their support for exporting Parquet with `DBMS_CLOUD.EXPORT_DATA`:

Oracle Type	Notes
BFILE	Not supported
BLOB	Not supported
DATE	Supported with the following limitation: DATE format supports only date, month and year. Hour, minute and seconds are not supported. See <code>DBMS_CLOUD</code> Package Oracle Data Type to Parquet Mapping for details on NLS format limitations for exporting DATE to Parquet.
INTERVAL DAY TO SECOND	Supported and is treated as string internally
INTERVAL YEAR TO MONTH	Supported and is treated as string internally
LONG	Not supported
LONG RAW	Not supported
NUMBER	Supported with the following limitations: – Can have maximum precision of 38 and scale equal to less than precision. – If no precision and scale is provided for the column NUMBER type, by default precision of 38 and scale of 20 is used. – Negative scale is not supported for NUMBER types.

Oracle Type	Notes
Object Types	Not supported
TIMESTAMP	Supported with the following limitations: – If there are multiple columns with different precision, highest precision will be taken. – <code>TIMESTAMP WITH TIME ZONE</code> Oracle datatype will use the timestamp only. See <code>DBMS_CLOUD</code> Package Oracle Data Type to Parquet Mapping for details on NLS format limitations for exporting <code>TIMESTAMP</code> to Parquet.

Export Data as XML to a Directory

Shows the steps to export table data from your Autonomous AI Database to Directory as XML data by specifying a query.

1. Connect to your Autonomous AI Database instance.

See [Connect to Autonomous AI Database](#) for more information.

2. Create a directory.

For example:

```
CREATE DIRECTORY export_dir AS 'export_dir';
```

See [Create a Directory](#) for more information.

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter type with the value `json` to export the results as XML files to a directory. Do not include the `credential` parameter when sending output to a directory.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'export_dir:sales.csv',
    format => JSON_OBJECT('type' value 'csv'),
    query => 'SELECT * FROM sales'
  );
END;
/
```

The directory name is case-sensitive when the directory name is enclosed in double quotes. For example:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => '"export_dir":sales.xml',
    format => JSON_OBJECT('type' value 'xml'),
    query => 'SELECT * FROM sales'
  );
END;
/
```

For detailed information about the parameters, see [EXPORT_DATA Procedure](#).

For detailed information about the available format parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see `DBMS_CLOUD` Package Format Options for `EXPORT_DATA`.

The parameters are:

- `file_uri_list`: is a comma delimited list of the export file(s). Use of wildcard and substitution characters is not supported in the `file_uri_list`.
- `format`: specifies the required type parameter. The valid values are `json`, `xml`, and `csv`.
- `query`: specifies a `SELECT` statement so that only the required data is exported. The query determines the contents of the dump file(s).

Note

The `DBMS_CLOUD.EXPORT_DATA` procedure creates the dump file(s) that you specify in the `file_uri_list`. The procedure does not overwrite files. If a dump file in the `file_uri_list` exists, `DBMS_CLOUD.EXPORT_DATA` generates another file with a unique name. `DBMS_CLOUD.EXPORT_DATA` does not create directories.

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files.

Export Data to Object Store as Text Using `DBMS_CLOUD.EXPORT_DATA`

Use `DBMS_CLOUD.EXPORT_DATA` to export data as text from an Autonomous AI Database to cloud Object Store. The text format export options are `CSV`, `JSON`, or `XML`.

Export JSON Data to Cloud Object Storage

Shows the steps to export table data from your Autonomous AI Database to Cloud Object Storage as JSON data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store, Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous AI Database instance.
See [Connect to Autonomous AI Database](#) for more information.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
```

```

        password => 'password'
    );
END;
/

```

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

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the `format` parameter type with the value `json` to export the results as JSON files on Cloud Object Storage.

To generate the JSON output files there are two options for the `file_uri_list` parameter:

- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage.
- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported JSON.

If you do not include the file name prefix in the `file_uri_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See [File Naming for Text Output \(CSV, JSON, Parquet, or XML\)](#) for details.

For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

```

BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'json'));
END;
/

```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

When record delimiters include escape characters, such as `\r\n` or `\t`, enclose the record delimiters in double quotes. For example, to use the record delimiter `\r\n`, enclose the value in double quotes: `"\r\n"`.

```

BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'json', 'recorddelimiter' value
'"\"r\n"' format json));
END;
/

```


In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

For detailed information about the parameters, see EXPORT_DATA Procedure.

For detailed information about the available format parameters you can use with DBMS_CLOUD.EXPORT_DATA, see DBMS_CLOUD Package Format Options for EXPORT_DATA.

Notes for exporting with DBMS_CLOUD.EXPORT_DATA:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files.
- Specify the format parameter with the encryption option to encrypt data while exporting. See [#unique_321](#) for more information.
- When you no longer need the files that you export, use the procedure DBMS_CLOUD.DELETE_OBJECT or use native Cloud Object Storage commands to delete the files.

Export Data as CSV to Cloud Object Storage

Shows the steps to export table data from your Autonomous AI Database to Cloud Object Storage as CSV data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous AI Database. You can also use Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous AI Database instance.
See [Connect to Autonomous AI Database](#) for more information.
2. Store your Cloud Object Storage credential using DBMS_CLOUD.CREATE_CREDENTIAL.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

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

3. Run DBMS_CLOUD.EXPORT_DATA and specify the format parameter type with the value csv to export the results as CSV files on Cloud Object Storage.

To generate the CSV output files there are two options for the file_uri_list parameter:

- Set the file_uri_list value to the URL for an existing bucket on your Cloud Object Storage.

- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported CSV files.

If you do not include the file name prefix in the `file_uri_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See [File Naming for Text Output \(CSV, JSON, Parquet, or XML\)](#) for details.

For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'csv', 'delimiter' value
'|', 'compression' value 'gzip'));
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

For detailed information about the available format parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see `DBMS_CLOUD` Package Format Options for `EXPORT_DATA`.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files.
- Specify the format parameter with the encryption option to encrypt data while exporting. See [#unique_321](#) for more information.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.

Export Data as Parquet to Cloud Object Storage

Shows the steps to export table data from your Autonomous AI Database to Cloud Object Storage as Parquet data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store, Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous AI Database instance.

See [Connect to Autonomous AI Database](#) for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

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

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the format parameter type with the value `parquet` to export the results as parquet files on Cloud Object Storage.

To generate the parquet output files there are two options for the `file_uri_list` parameter:

- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage.
- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported parquet files.

If you do not include the file name prefix in the `file_uri_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See [File Naming for Text Output \(CSV, JSON, Parquet, or XML\)](#) for details.

For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespaces-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'parquet', 'compression'
value 'snappy'));
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

For detailed information about the available format parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see `DBMS_CLOUD` Package Format Options for `EXPORT_DATA`.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files. The default compression for type parquet is snappy.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.
- See `DBMS_CLOUD` Package Oracle Data Type to Parquet Mapping for details on Oracle Type to Parquet Type mapping.

The following types are not supported or have limitations on their support for exporting Parquet with `DBMS_CLOUD.EXPORT_DATA`:

Oracle Type	Notes
BFILE	Not supported
BLOB	Not supported
DATE	Supported with the following limitation: DATE format supports only date, month and year. Hour, minute and seconds are not supported. See <code>DBMS_CLOUD</code> Package Oracle Data Type to Parquet Mapping for details on NLS format limitations for exporting DATE to Parquet.
INTERVAL DAY TO SECOND	Supported and is treated as string internally
INTERVAL YEAR TO MONTH	Supported and is treated as string internally
LONG	Not supported
LONG RAW	Not supported
NUMBER	Supported with the following limitations: – Can have maximum precision of 38 and scale equal to less than precision. – If no precision and scale is provided for the column NUMBER type, by default precision of 38 and scale of 20 is used. – Negative scale is not supported for NUMBER types.
Object Types	Not supported
TIMESTAMP	Supported with the following limitations: – If there are multiple columns with different precision, highest precision will be taken. – <code>TIMESTAMP WITH TIME_ZONE</code> Oracle datatype will use the timestamp only. See <code>DBMS_CLOUD</code> Package Oracle Data Type to Parquet Mapping for details on NLS format limitations for exporting TIMESTAMP to Parquet.

Export Data as XML to Cloud Object Storage

Shows the steps to export table data from your Autonomous AI Database to Cloud Object Storage as XML data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous AI Database. You can also use Amazon Resource Names (ARNs) to access AWS Simple Storage

Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous AI Database instance.
See [Connect to Autonomous AI Database](#) for more information.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

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

3. Run `DBMS_CLOUD.EXPORT_DATA` and specify the format parameter type with the value `xml` to export the results as XML files on Cloud Object Storage.

To generate the XML output files there are two options for the `file_uri_list` parameter:

- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage.
- Set the `file_uri_list` value to the URL for an existing bucket on your Cloud Object Storage and include a file name prefix to use when generating the file names for the exported JSON.

If you do not include the file name prefix in the `file_uri_list`, `DBMS_CLOUD.EXPORT_DATA` supplies a file name prefix. See [File Naming for Text Output \(CSV, JSON, Parquet, or XML\)](#) for details.

For example, the following shows `DBMS_CLOUD.EXPORT_DATA` with a file name prefix specified in `file_uri_list`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/dept_export',
    query          => 'SELECT * FROM DEPT',
    format         => JSON_OBJECT('type' value 'xml', 'compression' value
'gzip'));
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

For detailed information about the parameters, see `EXPORT_DATA` Procedure.

For detailed information about the available format parameters you can use with `DBMS_CLOUD.EXPORT_DATA`, see `DBMS_CLOUD` Package Format Options for `EXPORT_DATA`.

Notes for exporting with `DBMS_CLOUD.EXPORT_DATA`:

- The query parameter that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Specify the format parameter with the compression option to compress the output files.
- Specify the format parameter with the encryption option to encrypt data while exporting. See [#unique_321](#) for more information.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.

File Naming for Text Output (CSV, JSON, Parquet, or XML)

Describes the output file naming using `DBMS_CLOUD.EXPORT_DATA` with CSV, JSON, Parquet, or XML text file output.

`DBMS_CLOUD.EXPORT_DATA` performs the query specified with the query parameter and sends the results to text files either in the Cloud Object Store bucket or to a directory. The output format depends on the format parameter type you specify (one of CSV, JSON, Parquet, or XML).

To speed up the procedure and to generate the output as fast as possible, `DBMS_CLOUD.EXPORT_DATA` divides its work. This means that, depending on system resources, when you run `DBMS_CLOUD.EXPORT_DATA` the procedure creates multiple output files either in the Cloud Object Store bucket or in the directory.

The format for each generated file is:

[*FileNamePrefix* | *client_info_module_action*]_{sequenceNum_timestamp}.*format_extension*.
[*compression_extension*]

- *FileNamePrefix*: (optional) If a *FileNamePrefix* is supplied, `DBMS_CLOUD.EXPORT_DATA` uses the file name prefix to generate file names for the results. The *FileNamePrefix* is specified using the text supplied after the bucket or directory name in the *file_uri_list* parameter value.

You cannot provide multiple values for the *FileNamePrefix* in the *file_uri_list*.

- *client_info_module_action*: If a file name prefix is not supplied with the *file_uri_list* parameter, `DBMS_CLOUD.EXPORT_DATA` uses the combination of *client_info*, application *module* and *action* as the file name prefix (when this information is available). The procedure obtains these names from the application information for the database session that runs the query. See *DBMS_APPLICATION_INFO* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for information on *client_info*, *module* name, and *action* name.

If a file name prefix is not supplied with the *file_uri_list* and the database session attributes are not available, `DBMS_CLOUD.EXPORT_DATA` uses the file name prefix "data".

- *sequenceNum*: The sequence number associated with the `DBMS_CLOUD.EXPORT_DATA` query. Depending on the query, the database service, and the number of ECPU's (OCPU's if your database uses OCPU's) there are one or more *sequenceNums*. Also, depending on the size of the results, there are one or more output files for each *sequenceNum*.
- *timestamp*: Timestamp when the file is uploaded.

- *format_extension*: The default value depends on the format type value:
 - CSV format: .csv
 - JSON format: .json
 - PARQUET format .parquet
 - XML format: .xml

For more information, see the description for format option *fileextension* in DBMS_CLOUD Package Format Options for EXPORT_DATA.

- *compression_extension*: When you include the format parameter with the compression option with the value *gzip*, this is "gz".

When the format type is *parquet*, the compression value *snappy* is also supported and is the default.

For example, the file name prefix in the following DBMS_CLOUD.EXPORT_DATA procedure is specified in the *file_uri_list* parameter, as *dept_export*. The example generates the output to the provided Object Store in the specified format.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'json'));
END;
/
```

When you specify a file name prefix the generated output files include the file name prefix, similar to the following:

```
dept_export_1_20210809T173033Z.json
dept_export_2_20210809T173034Z.json
dept_export_3_20210809T173041Z.json
dept_export_4_20210809T173035Z.json
```

The number of generated output files depends on the size of the results, the database service, and the number of ECPU's (OCPUs if your database uses OCPUs) in the Autonomous AI Database instance.

In the following example the *file_uri_list* parameter does not include a file name prefix and the compression parameter is supplied, with value *gzip*:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/',
    query           => 'SELECT * FROM DEPT',
    format          => json_object('type' value 'json', 'compression' value
'gzip'));
END;
/
```


When a file name prefix is not in the `file_uri_list` parameter, `DBMS_CLOUD.EXPORT_DATA` uses a file name prefix of the form: *client_info_module_action*. For this example the generated output files include the file name prefix that `DBMS_CLOUD.EXPORT_DATA` supplies and the files are compressed with gzip and the file extension `.gz` is added, as follows:

```
Client1_Module1_Action1_1_20210809T173033Z.json.gz
Client1_Module1_Action1_2_20210809T173034Z.json.gz
Client1_Module1_Action1_3_20210809T173041Z.json.gz
Client1_Module1_Action1_4_20210809T173035Z.json.gz
```

If the *client_info_module_action* session information is not available when you run `DBMS_CLOUD.EXPORT_DATA`, the file name prefix is set to `data`. For example:

```
data_1_20210809T173033Z.json.gz
data_2_20210809T173034Z.json.gz
data_3_20210809T173041Z.json.gz
data_4_20210809T173035Z.json.gz
```

For example, the file name prefix in the following `DBMS_CLOUD.EXPORT_DATA` procedure is specified in the `file_uri_list` parameter, as `dept_export`. The example generates the output to the provided directory in the specified format.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list => 'DATA_PUMP_DIR:sales.json',
    query         => 'SELECT * FROM SALES',
    format        => JSON_OBJECT('type' value 'json'));
END;
/
```

When you specify a file name prefix the generated output file include the file name prefix, similar to the following:

```
sales_1_20230705T124523275915Z.csv
```

Notes for file naming with `DBMS_CLOUD.EXPORT_DATA`:

- `DBMS_CLOUD.EXPORT_DATA` does not create buckets or directories.
- The number of files that `DBMS_CLOUD.EXPORT_DATA` generates is determined by the number of ECPUs (OCPUs if your database uses OCPUs), the database service, and the size of the result data.
- The following applies when providing a directory object name in the `file_uri_list` parameter:
 - The provided directory must exist and you must have `WRITE` access to the directory.
 - The directory name is case-sensitive when enclosed in double quotes.
 - The credential name parameter must not be provided.
- For CSV, JSON, or XML output, by default when a generated file contains 10MB of data a new output file is created. However, if you have less than 10MB of result data you may have multiple output files, depending on the database service and the number of ECPUs (OCPUs if your database uses OCPUs) for the Autonomous Database instance.

The default output file chunk size is 10MB for CSV, JSON, or XML. You can change this value with the format parameter `maxfilesize` option. See `DBMS_CLOUD` Package Format Options for `EXPORT_DATA` for more information.

- For Parquet output, each generated file is less than 128MB and multiple output files may be generated. However, if you have less than 128MB of result data, you may have multiple output files depending on the database service and the number of ECPU (OCPU if your database uses OCPU) for the Autonomous Database instance.

The format parameter `maxfilesize` option does not apply for Parquet files.

Encrypt Data While Exporting to Object Storage

You can encrypt table data while exporting to Object Storage.

Use the format parameter and the encryption option with `DBMS_CLOUD.EXPORT_DATA` to encrypt data when you export from Autonomous AI Database to Object Storage.

Note the following when you export encrypted data to Object Storage:

- The encryption option is only supported when exporting data from Autonomous AI Database to Object Storage as CSV, JSON, or XML.
- When the export includes both encryption and compression, the order of operations is: first the data is compressed, next the data is encrypted, and then it is uploaded to Object Storage.
- There are two supported encryption methods:
 - Using a user-defined function.
 - Using a `DBMS_CCRYPTO` specified encryption algorithm.

See [DBMS_CCRYPTO](#) for information on the cryptographic functions and procedures for encryption and decryption.

Encrypt Data Using DBMS_CCRYPTO Encryption Algorithms

Shows the steps to encrypt data using `DBMS_CCRYPTO` encryption algorithms while exporting to Cloud Object Storage.

Perform the following steps to encrypt data while exporting to Cloud Object Storage (this example exports table data to a CSV file):

1. Connect to your Autonomous AI Database instance.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OBJ_STORE_CRED',
    username        => 'user1@example.com',
    password        => 'password'
  );
END;
/
```

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

See CREATE_CREDENTIAL Procedure for more information.

3. Create a credential to store the encryption key (the encryption key to be used for encrypting data).

When you encrypt data using DBMS_CRYPT0 encryption algorithms you store the encryption key in a credential. The key is specified in the password field in a credential you create with DBMS_CLOUD.CREATE_CREDENTIAL.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'ENC_CRED_NAME',
    username       => 'Any_username',
    password       => 'password'
  );
END;
/
```

4. Run DBMS_CLOUD.EXPORT_DATA.

Use the format parameter with the encryption option. The encryption type specifies the DBMS_CRYPT0 encryption algorithm to use to encrypt the table data and the credential_name value is credential that specifies the secret (encryption key).

For example:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA (
    credential_name => 'OBJ_STORE_CRED',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/encrypted.csv',
    query           => 'SELECT * FROM ADMIN.employees',
    format          => json_object(
      'type' value 'csv',
      'encryption' value json_object(
        'type' value DBMS_CRYPT0.ENCRYPT_AES256 +
DBMS_CRYPT0.CHAIN_CBC + DBMS_CRYPT0.PAD_PKCS5,
        'credential_name' value 'ENC_CRED_NAME'))
    );
END;
/
```

This encrypts and exports the data from the EMPLOYEES table into a CSV file.

See [DBMS_CRYPT0 Algorithms](#) for more information on encryption algorithms.

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

See EXPORT_DATA Procedure and DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.

After you encrypt files with DBMS_CLOUD.EXPORT_DATA, when you use DBMS_CRYPT0 encryption algorithms to encrypt the files, you have these options for using or importing the files you exported:

- You can use `DBMS_CLOUD.COPY_DATA` or `DBMS_CLOUD.COPY_COLLECTION` with the same encryption algorithm options and the key to decrypt the files.

See [Decrypt and Load Data Using DBMS_CCRYPTO Algorithms](#) for more information.

- You can query the data in an external table by supplying the same encryption algorithm options and the key to decrypt the files, with any of the following procedures:

- `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`
- `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`
- `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE`

For `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` this option is only applicable to the Object Storage files.

See [Decrypt and Load Data Using DBMS_CCRYPTO Algorithms](#) for more information.

- On a system that is not an Autonomous AI Database you can use the `DBMS_CCRYPTO` package with the same algorithm options and the key to decrypt the files.

Note that the key is stored as a `VARCHAR2` in the credential in Autonomous AI Database but `DBMS_CCRYPTO` uses `RAW` type for the key parameter.

See [DBMS_CCRYPTO Algorithms](#) for more information on encryption algorithms.

Encrypt Data with a User Defined Encryption Function

Shows the steps to encrypt data using a user-defined encryption function while exporting to Cloud Object Storage.

Perform the following steps to encrypt data while exporting to Cloud Object Storage (this example exports table data to a CSV file):

1. Connect to your Autonomous AI Database instance.
2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OBJ_STORE_CRED',
    username       => 'user1@example.com',
    password       => 'password'
  );
END;
/
```

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

See [CREATE_CREDENTIAL Procedure](#) for more information.

3. Create a user-defined callback function to encrypt data.

For example:

```
CREATE OR REPLACE FUNCTION encryption_func (data IN BLOB)
RETURN BLOB
IS
  l_encrypted_data BLOB;
```

```

BEGIN
  DBMS_LOB.CREATETEMPORARY (l_encrypted_data, TRUE, DBMS_LOB.CALL);
  DBMS_CRYPTO.ENCRYPT (
    dst => l_encrypted_data,
    src => data,
    typ => DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC +
    DBMS_CRYPTO.PAD_PKCS5,
    key => 'encryption key'
  );
  RETURN l_encrypted_data;
END encryption_func;
/

```

This creates the ENCRYPTION_FUNC encryption function. This function encrypts data using a stream or block cipher with a user supplied key.

Note

You must create an encryption key to be used as a value in the KEY parameter. See [DBMS_CRYPTO Operational Notes](#) for more information on generating the encryption key.

4. Run DBMS_CLOUD.EXPORT_DATA with the format parameter, include the encryption option and specify a user_defined_function.

For example:

```

BEGIN
  DBMS_CLOUD.EXPORT_DATA (
    credential_name => 'OBJ_STORE_CRED',
    file_uri_list   => 'https://objectstorage.us-
    phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/encrypted.csv',
    query           => 'SELECT * FROM ADMIN.emp',
    format          => json_object(
                        'type' value 'csv',
                        'encryption' value
    json_object('user_defined_function' value 'admin.encryption_func'))
  );
END;
/

```

This encrypts the data from the specified query the on EMP table and exports the data as a CSV file on Cloud Object Storage. The format parameter with the encryption value specifies the user-defined encryption function to use to encrypt the data.

Note

You must have EXECUTE privilege on the encryption function.

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

See `EXPORT_DATA` Procedure and `DBMS_CLOUD` Package Format Options for `EXPORT_DATA` for more information.

Use Data Pipelines for Continuous Load and Export

About Data Pipelines on Autonomous AI Database on Dedicated Exadata Infrastructure

Autonomous AI Database data pipelines are either load pipelines or export pipelines.

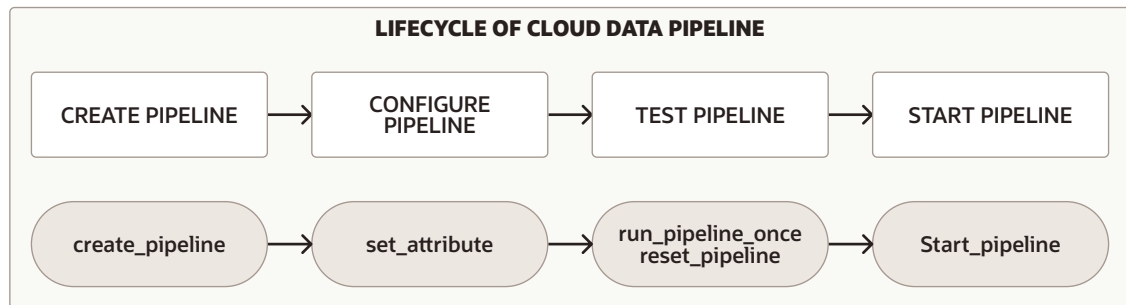
Load pipelines provide continuous incremental data loading from external sources (as data arrives on object store it is loaded to a database table). Export pipelines provide continuous incremental data exporting to object store (as new data appears in a database table it is exported to object store). Pipelines use database scheduler to continuously load or export incremental data.

Autonomous AI Database data pipelines provide the following:

- **Unified Operations:** Pipelines allow you to quickly and easily load or export data and repeat these operations at regular intervals for new data. The `DBMS_CLOUD_PIPELINE` package provides a unified set of PL/SQL procedures for pipeline configuration and for creating and starting a scheduled job for load or export operations.
- **Scheduled Data Processing:** Pipelines monitor their data source and periodically load or export the data as new data arrives.
- **High Performance:** Pipelines scale data transfer operations with the available resources on your Autonomous AI Database. Pipelines by default use parallelism for all load or export operations, and scale based on the CPU resources available on your Autonomous AI Database, or based on a configurable priority attribute.
- **Atomicity and Recovery:** Pipelines guarantee atomicity such that files in object store are loaded exactly once for a load pipeline.
- **Monitoring and Troubleshooting:** Pipelines provide detailed log and status tables that allow you to monitor and debug pipeline operations.
- **Multicloud Compatible:** Pipelines on Autonomous AI Database support easy switching between cloud providers without application changes. Pipelines support all the credential and object store URI formats that Autonomous AI Database supports (Oracle Cloud Infrastructure Object Storage, Amazon S3, Azure Blob Storage, Google Cloud Storage, and Amazon S3-Compatible object stores).

Data Pipeline Lifecycle

The `DBMS_CLOUD_PIPELINE` package provides procedures for creating, configuring, testing, and starting a pipeline. The pipeline lifecycle and procedures are the same for both load and export pipelines.



For either pipeline type you perform the following steps to create and use a pipeline:

1. Create and configure the pipeline. See [Create and Configure Pipelines](#) for more information.
2. Test a new pipeline. See [Test Pipelines](#) for more information.
3. Start a pipeline. See [Start a Pipeline](#) for more information.

In addition, you can monitor, stop, or drop pipelines:

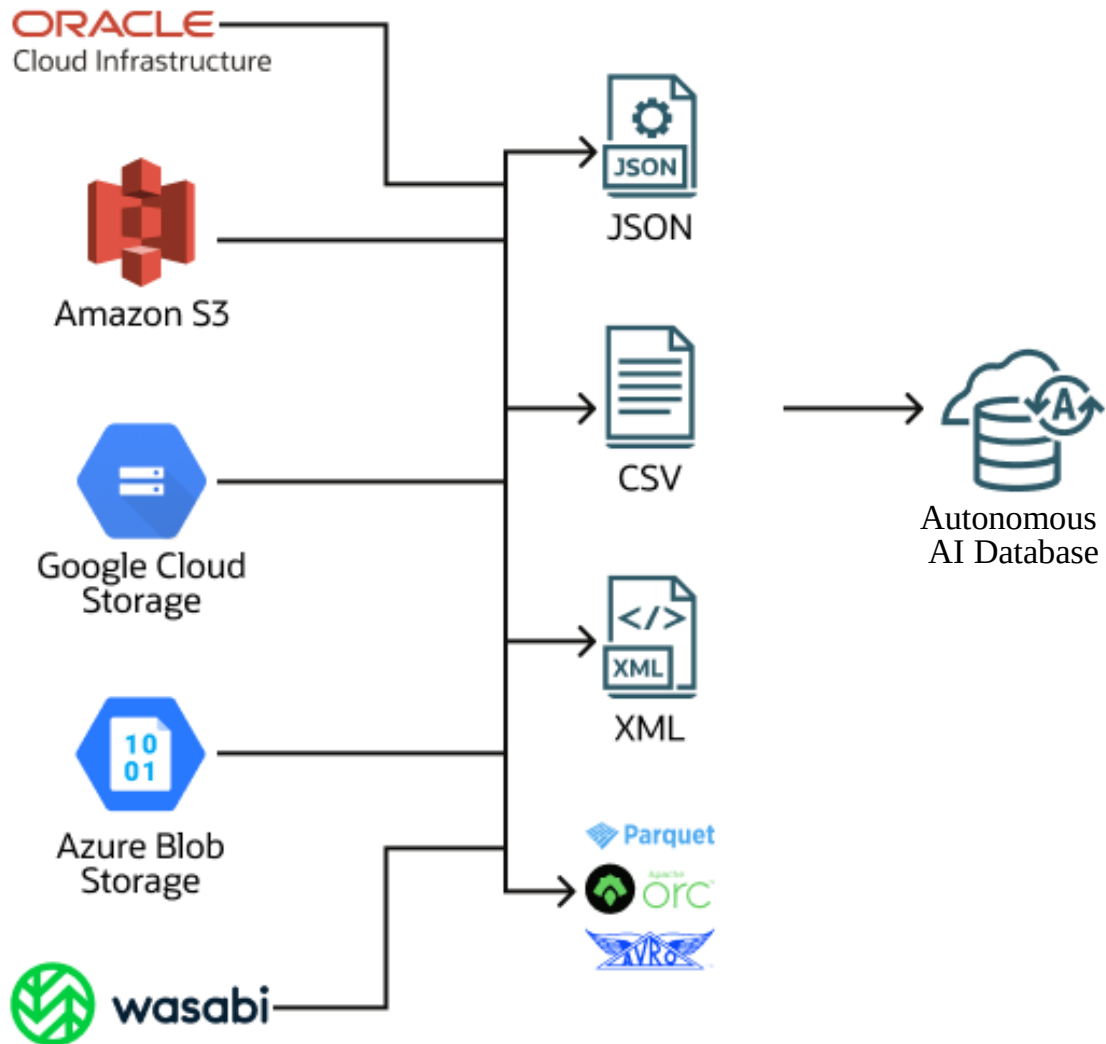
- While a pipeline is running, either during testing or during regular use after you start the pipeline, you can monitor the pipeline. See [Monitor and Troubleshoot Pipelines](#) for more information.
- You can stop a pipeline and later start it again, or drop a pipeline when you are finished using the pipeline. See [Stop a Pipeline and Drop a Pipeline](#) for more information.

Load Pipelines

Use a load pipeline for continuous incremental data loading from external files in object store into a database table. A load pipeline periodically identifies new files in object store and loads the new data into the database table.

A load pipeline operates as follows (some of these features are configurable using pipeline attributes):

- Object store files are loaded in parallel into a database table.
 - A load Pipeline uses the object store file name to uniquely identify and load newer files.
 - Once a file in object store has been loaded in the database table, if the file content changes in object store, it will not be loaded again.
 - If the object store file is deleted, it does not impact the data in the database table.
- If failures are encountered, a load pipeline automatically retries the operation. Retries are attempted on every subsequent run of the pipeline's scheduled job.
- In cases where the data in a file does not comply with the database table, it is marked as FAILED and can be reviewed to debug and troubleshoot the issue.
 - If any file fails to load, the pipeline does not stop and continues to load the other files.
- Load pipelines support multiple input file formats, including: JSON, CSV, XML, Avro, ORC, and Parquet.



Migration from non-Oracle databases is one possible use case for a load pipeline. When you need to migrate your data from a non-Oracle database to Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, you can extract the data and load it into Autonomous AI Database (Oracle Data Pump format cannot be used for migrations from non-Oracle databases). By using a generic file format such as CSV to export data from a non-Oracle database, you can save your data to files and upload the files to object store. Next, create a pipeline to load the data to Autonomous AI Database. Using a load pipeline to load a large set of CSV files provides important benefits such as fault tolerance, and resume and retry operations. For a migration with a large data set you can create multiple pipelines, one per table for the non-Oracle database files, to load data into Autonomous AI Database.

Export Pipelines

Use an export pipeline for continuous incremental export of data from the database to object store. An export pipeline periodically identifies candidate data and uploads the data to object store.

There are three export pipeline options (the export options are configurable using pipeline attributes):

- Export incremental results of a query to object store using a date or timestamp column as key for tracking newer data.

- Export incremental data of a table to object store using a date or timestamp column as key for tracking newer data.
- Export data of a table to object store using a query to select data without a reference to a date or timestamp column (so that the pipeline exports all the data that the query selects for each scheduler run).

Export pipelines have the following features (some of these are configurable using pipeline attributes):

- Results are exported in parallel to object store.
- In case of any failures, a subsequent pipeline job repeats the export operation.
- Export pipelines support multiple export file formats, including: CSV, JSON, Parquet, or XML.

Oracle Maintained Pipelines

Autonomous AI Database on Dedicated Exadata Infrastructure provides built-in pipelines to export specific logs to an Object Store in JSON format. These pipelines are preconfigured and are started and owned by the ADMIN user.

The Oracle Maintained pipelines are:

- **ORA\$AUDIT_EXPORT**: This pipeline exports the database audit logs to object store in JSON format and runs every 15 minutes after starting the pipeline (based on the `interval` attribute value).
- **ORA\$APEX_ACTIVITY_EXPORT**: This pipeline exports the Oracle APEX workspace activity log to object store in JSON format. This pipeline is preconfigured with the SQL query for retrieving APEX activity records and runs every 15 minutes after starting the pipeline (based on the `interval` attribute value).

To configure and start an Oracle Managed pipeline:

1. Determine the Oracle Managed Pipeline you want to use: **ORA\$AUDIT_EXPORT** or **ORA\$APEX_ACTIVITY_EXPORT**.
2. Set the `credential_name` and `location` attributes.

Note

The `credential_name` is a mandatory value on Autonomous AI Database on Dedicated Exadata Infrastructure.

For example:

```
BEGIN
  DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
    pipeline_name => 'ORA$AUDIT_EXPORT',
    attribute_name => 'credential_name',
    attribute_value => 'DEF_CRED_OBJ_STORE'
  );
  DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
    pipeline_name => 'ORA$AUDIT_EXPORT',
    attribute_name => 'location',
    attribute_value => 'https://objectstorage.us-
    phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/'
  );
END;
```



```
);
END;
/
```

The log data from database is exported to the object store location you specify.

See `SET_ATTRIBUTE` for more information.

3. Optionally, set the interval, format, or priority attributes.

See `SET_ATTRIBUTE` for more information.

4. Start the pipeline.

See `START_PIPELINE` for more information.

Create and Configure Pipelines

You can create one or more load or export pipelines. When you create a pipeline, you use parameters and set pipeline attributes to configure the pipeline.

The options to create and configure a pipeline follow:

- **Load Pipeline:**
 - See [Create and Configure a Pipeline for Loading Data](#).
- **Export Pipeline:**
 - Export incremental results of a query to object store using a date or timestamp column as key for tracking newer data. See [Create and Configure a Pipeline for Export with Timestamp Column](#).
 - Export incremental data of a table to object store using a date or timestamp column as key for tracking newer data. See [Create and Configure a Pipeline for Export with Timestamp Column](#).
 - Export data of a table to object store using a query to select data without a reference to a date or timestamp column (so that the pipeline exports all the data that the query selects for each scheduler run). See [Create and Configure a Pipeline to Export Query Results \(Without a Timestamp\)](#).

Create and Configure a Pipeline for Loading Data

You can create a pipeline to load data from external files in object store or directories to tables in Autonomous AI Database.

A load pipeline consumes data placed on object store or in directories and loads it to a table in Autonomous AI Database. When you create a load pipeline, the pipeline runs at regular intervals to consume data placed on the source location, when new data files arrive the pipeline loads the new data. You can also use a pipeline to reliably copy files, with resume and retry capabilities, from the source location to a table on your database.

With a load pipeline, the pipeline package uses `DBMS_CLOUD.COPY_DATA` to load data.

On your Autonomous AI Database, either use an existing table or create the database table where you are loading data. For example:

```
CREATE TABLE EMPLOYEE
      (name      VARCHAR2(128),
```

```

age      NUMBER,
salary   NUMBER);

```

1. Create a pipeline to load data from object store or from directory objects.

```

BEGIN
  DBMS_CLOUD_PIPELINE.CREATE_PIPELINE(
    pipeline_name => 'MY_PIPE1',
    pipeline_type => 'LOAD',
    description   => 'Load metrics from object store into a table'
  );
END;
/

```

See CREATE_PIPELINE Procedure for more information.

2. Create a credential object to access the object store that contains the files you are loading.

You specify the credential for the pipeline source location with the attribute `credential_name`. If you do not supply a `credential_name` in the next step, the `credential_name` value is set to NULL. You can use the default NULL value when the location attribute is a public or pre-authenticated URL.

See CREATE_CREDENTIAL Procedure for more information.

3. Set the pipeline attributes, including the required attributes: location, table_name, and format.

Case 1: Create a pipeline for loading data from object store.

```

BEGIN
  DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
    pipeline_name => 'MY_PIPE1',
    attributes    => JSON_OBJECT(
      'credential_name' VALUE 'OBJECT_STORE_CRED',
      'location' VALUE 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
      'table_name' VALUE 'employee',
      'format' VALUE '{"type":"json", "columnpath":["$.NAME",
"$$.AGE", "$$.SALARY"]}',
      'priority' VALUE 'HIGH',
      'interval' VALUE '20')
    );
END;
/

```

Case 2: Create a pipeline for loading data from directory objects.

```

BEGIN
  DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
    pipeline_name => 'MY_PIPE1',
    pipeline_type => 'LOAD',
    attributes    => JSON_OBJECT(
      'location' VALUE 'MY_DIR:*.csv',
      'table_name' VALUE 'employee',

```

```

        'format' VALUE '{"type": "csv"}',
        'priority' VALUE 'HIGH',
        'interval' VALUE '20')
    );
END;
/

```

The following attributes must be set to run a load pipeline in both cases:

- **location:** Specifies the source file location on object store or directory object.
- **table_name:** Specifies the table in your database where you are loading data. The location you specify is for one table_name per pipeline.
- **format:** Describes the format of the data you are loading.

See DBMS_CLOUD Package Format Options for more information.

The credential_name is the credential you created in the previous step.

The priority value determines the number of files loaded in parallel. A pipeline with a higher priority consumes more database resources and completes each run faster, as compared to running at a lower priority.

The interval value specifies the time interval in minutes between consecutive runs of a pipeline job. The default interval is 15 minutes.

See DBMS_CLOUD_PIPELINE Attributes for details on the pipeline attributes.

4. After you create a pipeline you can test the pipeline or start the pipeline:

- [Test Pipelines](#)
- Start a Pipeline

As an alternative, to set the format for JSON, you could use the following format:

```

BEGIN
    DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
        pipeline_name => 'MY_PIPE1',
        attribute_name => 'format',
        attribute_value => JSON_OBJECT('type' value 'json', 'columnpath'
value '["$$.NAME", "$$.AGE", "$$.SALARY"]')
    );
END;
/

```

Create and Configure a Pipeline for Export with Timestamp Column

You can create an export pipeline to automatically export time-series data from your Autonomous AI Database to object store.

Using this export pipeline option you specify a table or SQL query and a column with a timestamp that the pipeline uses to keep track of the time of the last upload. You can use an export pipeline to share data for consumption by other applications or to save data to object store.

With an export pipeline, the pipeline package uses DBMS_CLOUD.EXPORT_DATA to export data.

An export pipeline exports data from your Autonomous AI Database to object store. When you create an export pipeline, the pipeline runs at regular intervals and places data on object store.

1. Create a pipeline to export data to object store.

```
BEGIN
    DBMS_CLOUD_PIPELINE.CREATE_PIPELINE(
        pipeline_name=>'EXP_PIPE1',
        pipeline_type=>'EXPORT',
        description=>'Export time series metrics to object store');
END;
/
```

See CREATE_PIPELINE Procedure for more information.

2. Create a credential object to access the destination object store location where you are exporting data files.

You specify the credential for the pipeline destination location with the attribute `credential_name`. If you do not supply a `credential_name` in the next step, the `credential_name` value is set to NULL. You can use the default NULL value when the location attribute is a public or pre-authenticated URL.

See CREATE_CREDENTIAL Procedure for more information.

3. Set the export pipeline attributes.

When you specify a `table_name` parameter, table rows are exported to object store. When you specify a `query` parameter, the query specifies a SELECT statement so that only the required data is exported to object store.

- Using using a `table_name` parameter:

```
BEGIN
    DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
        pipeline_name => 'EXP_PIPE1',
        attributes     => JSON_OBJECT('credential_name' VALUE
'OBJECT_STORE_CRED',
        'location' VALUE 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
        'table_name' VALUE 'metric_table',
        'key_column' VALUE 'metric_time',
        'format' VALUE '{"type": "json"}',
        'priority' VALUE 'MEDIUM',
        'interval' VALUE '20')
    );
END;
/
```

- Using a query parameter:

```
BEGIN
    DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
        pipeline_name => 'EXP_PIPE1',
        attributes     => JSON_OBJECT('credential_name' VALUE
'OBJECT_STORE_CRED',
        'location' VALUE 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
        'query' VALUE 'SELECT * from metrics_table',
        'key_column' VALUE 'metric_time',
        'format' VALUE '{"type": "json"}',
```

```

        'priority' VALUE 'MEDIUM',
        'interval' VALUE '20')
    );
END;
/

```

Where the `credential_name` is the credential you created in the previous step.

The following attributes must be set to run an export pipeline:

- `location`: Specifies the destination object store location. The `location` you specify is for one `table_name` per pipeline.
- `table_name`: Specifies the table in your database that contains the data you are exporting (either the `table_name` parameter or the `query` parameter is required).
- `query`: Specifies the query to run in your database that provides the data you are exporting (either the `table_name` parameter or the `query` parameter is required).
- `format`: Describes the format of the data you are exporting.

See DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.

The `priority` value determines the degree of parallelism for fetching data from the database.

The `interval` value specifies the time interval in minutes between consecutive runs of a pipeline job. The default interval is 15 minutes.

See DBMS_CLOUD_PIPELINE Attributes for details on the pipeline attributes.

After you create a pipeline you can test the pipeline or start the pipeline:

- [Test Pipelines](#)
- Start a Pipeline

Create and Configure a Pipeline to Export Query Results (Without a Timestamp)

You can create an export pipeline to automatically export data from your Autonomous AI Database to object store. Using this export pipeline option you specify a SQL query that the pipeline runs periodically to export data to object store. You can use this export option to share the latest data from your Autonomous AI Database to object store for other applications to consume the data.

An export pipeline exports data from your Autonomous AI Database to object store. When you create an export pipeline, the pipeline runs at regular intervals and places data on object store.

1. Create a pipeline to export data to object store.

```

BEGIN
    DBMS_CLOUD_PIPELINE.CREATE_PIPELINE(
        pipeline_name=>'EXP_PIPE2',
        pipeline_type=>'EXPORT',
        description=>'Export query results to object store. ');
END;
/

```

See CREATE_PIPELINE Procedure for more information.

2. Create a credential object to access the destination object store location where you are exporting data files.

You specify the credential for the pipeline destination location with the attribute `credential_name`. If you do not supply a `credential_name` in the next step, the `credential_name` value is set to NULL. You can use the default NULL value when the location attribute is a public or pre-authenticated URL.

See `CREATE_CREDENTIAL` Procedure for more information.

3. Set the export pipeline attributes.

```
BEGIN
    DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(
        pipeline_name => 'EXP_PIPE2',
        attributes     => JSON_OBJECT(
            'credential_name' VALUE 'OBJECT_STORE_CRED',
            'location' VALUE 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
            'query' VALUE 'SELECT * FROM table_name',
            'format' VALUE '{"type": "json"}',
            'priority' VALUE 'MEDIUM',
            'interval' VALUE '20')
    );
END;
/
```

Where the `credential_name` is the credential you created in the previous step.

The following attributes must be set to run an export pipeline:

- `location`: Specifies the destination object store location.
- `query`: Specifies the query to run in your database that provides the data you are exporting.
- `format`: Describes the format of the data you are exporting.

See `DBMS_CLOUD` Package Format Options for `EXPORT_DATA` for more information.

The `priority` value determines the degree of parallelism for fetching data from the database.

The `interval` value specifies the time interval in minutes between consecutive runs of a pipeline job. The default interval is 15 minutes.

See `DBMS_CLOUD_PIPELINE` Attributes for details on the pipeline attributes.

After you create a pipeline you can test the pipeline or start the pipeline:

- [Test Pipelines](#)
- Start a Pipeline

Test Pipelines

Use `RUN_PIPELINE_ONCE` to run a pipeline once on-demand without creating a scheduled job.

`RUN_PIPELINE_ONCE` is useful for testing a pipeline before you start the pipeline. After you run a pipeline once to test the pipeline and check that it is working as expected, use `RESET_PIPELINE` to reset the pipeline's state (to the state before you ran `RUN_PIPELINE_ONCE`).

1. Create a pipeline.

See [Create and Configure a Pipeline for Loading Data](#) for more information.

2. Run a pipeline once to test the pipeline.

```
BEGIN
    DBMS_CLOUD_PIPELINE.RUN_PIPELINE_ONCE(
        pipeline_name => 'MY_PIPE1'
    );
END;
/
```

See RUN_PIPELINE_ONCE Procedure for more information.

3. Perform any required checks to verify the pipeline is operating as expected.

See Monitor and Troubleshoot Pipelines for more information.

4. Reset the pipeline.

```
BEGIN
    DBMS_CLOUD_PIPELINE.RESET_PIPELINE(
        pipeline_name => 'MY_PIPE1',
        purge_data => TRUE
    );
END;
/
```

See RESET_PIPELINE Procedure for more information.

Control Pipelines (Start, Stop, Drop, or Reset a Pipeline)

After you create and test a pipeline, you control a pipeline by starting, stopping, or dropping the pipeline. You can also reset a pipeline.

Start a Pipeline

After you create a pipeline, you can start the pipeline.

When a pipeline is started the pipeline runs continuously in a scheduled job. The pipeline's scheduled job repeats, either by default every 15 minutes or at the interval you set with the `interval` attribute.

1. Start a pipeline.

```
BEGIN
    DBMS_CLOUD_PIPELINE.START_PIPELINE(
        pipeline_name => 'EMPLOYEE_PIPELINE'
    );
END;
/
```

By default, a pipeline job begins immediately, as soon as the pipeline is started. To start a pipeline job at a later time, specify a valid future date or timestamp using the `start_date` parameter.

See `START_PIPELINE` Procedure for more information.

2. Verify that the pipeline is started.

For example:

```
SELECT pipeline_name, status from USER_CLOUD_PIPELINES
       WHERE pipeline_name = 'EMPLOYEE_PIPELINE';
```

PIPELINE_NAME	STATUS
-----	-----
EMPLOYEE_PIPELINE	STARTED

Stop a Pipeline

Use `STOP_PIPELINE` to stop a pipeline. When a pipeline is stopped, no future jobs are scheduled for the pipeline.

By default currently running jobs complete when you stop a pipeline. Set the force parameter to `TRUE` to terminate any running jobs and stop the pipeline immediately.

1. Stop a pipeline.

```
BEGIN
  DBMS_CLOUD_PIPELINE.STOP_PIPELINE(
    pipeline_name => 'EMPLOYEE_PIPELINE'
  );
END;
/
```

See `STOP_PIPELINE` Procedure for more information.

2. Verify that the pipeline is stopped.

```
SELECT pipeline_name, status from USER_CLOUD_PIPELINES
       WHERE pipeline_name = 'EMPLOYEE_PIPELINE';
```

PIPELINE_NAME	STATUS
-----	-----
EMPLOYEE_PIPELINE	STOPPED

See `STOP_PIPELINE` Procedure for more information.

Drop a Pipeline

The procedure `DROP_PIPELINE` drops an existing pipeline.

If a pipeline has been started, it must be stopped before the pipeline can be dropped. See `STOP_PIPELINE` Procedure for more information.

In order to drop a pipeline that is started, set the force parameter to `TRUE` to terminate any running jobs and drop the pipeline immediately

1. Drop a pipeline.

```
BEGIN
  DBMS_CLOUD_PIPELINE.DROP_PIPELINE(
    pipeline_name => 'EMPLOYEE_PIPELINE'
  );
END;
/
```

2. Verify the pipeline is dropped.

```
SELECT pipeline_name, status from USER_CLOUD_PIPELINES
WHERE pipeline_name = 'EMPLOYEE_PIPELINE';
```

No rows selected

See DROP_PIPELINE Procedure for more information.

Reset a Pipeline

Use the reset pipeline operation to clear the record of the pipeline to the initial state.

Note

You can optionally use reset pipeline to purge data in the database table associated with a load pipeline or to remove files in object store for an export pipeline. Usually this option is used when you are testing a pipeline during pipeline development.

Reset pipeline operates as follows:

- **Load Pipeline:** For a load pipeline, resetting the pipeline clears the record of the files being loaded by the pipeline. When you call either START_PIPELINE or RUN_PIPELINE_ONCE after resetting a load pipeline, the pipeline repeats the data load and includes all the files present in the object store location.

When purge_data is set to TRUE, the procedure truncates the data in the database table.

- **Export Pipeline:** For an export pipeline, resetting the pipeline clears the last tracked data in the database table. When you call either START_PIPELINE or RUN_PIPELINE_ONCE after resetting an export pipeline, the pipeline repeats exporting data from the table or query.

When purge_data set to TRUE, the procedure deletes existing files in the object store location specified with the location attribute.

To reset a pipeline:

1. Stop the pipeline you want to reset.

A data pipeline must be in stopped state to reset it. See STOP_PIPELINE Procedure for more information.

2. Reset the pipeline.

```
BEGIN
  DBMS_CLOUD_PIPELINE.RESET_PIPELINE(
    pipeline_name => 'EMPLOYEE_PIPELINE',
    purge_data => TRUE);
END;
```

```
END;
/
```

Only use the `purge_data` parameter with value `TRUE` if you want to clear data in your database table, for a load pipeline, or clear files in object store for an export pipeline. Usually this option is used when you are testing a pipeline during pipeline development.

See `RESET_PIPELINE` Procedure for more information.

Monitor and Troubleshoot Pipelines

All pipelines that are created are logged in the `DBMS_CLOUD_PIPELINE` views.

View Pipeline Status Information

Check pipeline status and obtain other pipeline information using the `USER_CLOUD_PIPELINES` or `DBA_CLOUD_PIPELINES` views. For example, the following `SELECT` statement with a `WHERE` clause predicate on the `pipeline_name` shows that `MY_TREE_DATA` is a load pipeline and the pipeline is started:

```
SELECT pipeline_name, pipeline_type, status FROM USER_CLOUD_PIPELINES
       WHERE pipeline_name = 'MY_TREE_DATA';
```

```
PIPELINE_NAME PIPELINE_TYPE STATUS
-----
MY_TREE_DATA  LOAD          STARTED
```

See `DBMS_CLOUD_PIPELINE` Views for more information.

View Pipeline Attributes

The pipeline attributes can be monitored by querying the `USER_CLOUD_PIPELINE_ATTRIBUTES` or `DBA_CLOUD_PIPELINE_ATTRIBUTES` views. Query these views to see pipeline attribute information.

For example:

```
SELECT pipeline_name, attribute_name, attribute_value FROM
user_cloud_pipeline_attributes
       WHERE pipeline_name = 'MY_TREE_DATA';
```

```
PIPELINE_NAME ATTRIBUTE_NAME
ATTRIBUTE_VALUE
```

```
-----
---
MY_TREE_DATA credential_name
DEF_CRED_OBJ_STORE

MY_TREE_DATA format          {"type":
"csv"}
MY_TREE_DATA interval
20
```

```
MY_TREE_DATA location      https://objectstorage.us-
ashburn-1.oraclecloud.com/n/namespaces/b/treetypes/o/
MY_TREE_DATA priority
high
```

```
MY_TREE_DATA table_name    TREES
```

See DBMS_CLOUD_PIPELINE Views for more information.

View Pipeline History

The USER_CLOUD_PIPELINE_HISTORY and DBA_CLOUD_PIPELINE_HISTORY views show the state of running jobs. Use the pipeline history views to help you monitor the health of a pipeline and detect failures in a running pipeline.

For example:

```
SELECT pipeline_id, pipeline_name, status, error_message FROM
user_cloud_pipeline_history
WHERE pipeline_name = 'MY_TREE_DATA';
```

```
PIPELINE_ID PIPELINE_NAME STATUS    ERROR_MESSAGE
-----
7 MY_TREE_DATA SUCCEEDED
```

See DBMS_CLOUD_PIPELINE Views for more information.

Pipeline Status Table: Additional Monitoring for Load Pipelines

The pipeline status table shows each file name and its status for a load pipeline. The STATUS_TABLE column in DBA_CLOUD_PIPELINES and USER_CLOUD_PIPELINES shows the status table name.

For example, the following SELECT statement with a WHERE clause predicate on the pipeline_name shows the status table name for a pipeline:

```
SELECT pipeline_name, status_table FROM user_cloud_pipelines
WHERE pipeline_name = 'MY_TREE_DATA';
```

```
PIPELINE_NAME STATUS_TABLE
-----
MY_TREE_DATA PIPELINE$9$41_STATUS
```

View the status table to see information about the pipeline, including the following:

- The relevant error number and error message are recorded in the status table if an operation on a specific file fails.
- For completed pipeline operations, the time needed for each operation can be calculated using the reported START_TIME and END_TIME.

For example the following shows that the load operation for two files failed and one completed:

```
SELECT id, name, status, error_code, error_message, sid FROM
PIPELINE$9$41_STATUS;
```

```
ID NAME          STATUS  ERROR_CODE ERROR_MESSAGE          SID
```

```

-----
1 trees1.txt FAILED          30653 ORA-30653: reject limit reached 18070
2 trees2.txt FAILED          30653 ORA-30653: reject limit reached 18070
3 trees3.txt COMPLETED                                18070

```

Pipelines for loading data, where the pipeline_type is 'LOAD', reserve an ID that is shown in USER_LOAD_OPERATIONS and in DBA_LOAD_OPERATIONS. The ID value in these views maps to the pipeline's OPERATION_ID in USER_CLOUD_PIPELINES and DBA_CLOUD_PIPELINES.

To obtain more information for a load pipeline, query the pipeline's OPERATION_ID:

```

SELECT PIPELINE_NAME, OPERATION_ID FROM USER_CLOUD_PIPELINES
       WHERE PIPELINE_NAME = 'MY_TREE_DATA';

```

```

PIPELINE_NAME OPERATION_ID
-----
MY_TREE_DATA      41

```

Next, query either USER_LOAD_OPERATIONS or DBA_LOAD_OPERATIONS with a WHERE clause predicate on the ID column (using the OPERATION_ID value).

For example:

```

SELECT ID, TYPE, LOGFILE_TABLE, BADFILE_TABLE, STATUS_TABLE FROM
USER_LOAD_OPERATIONS
       WHERE ID = 41;

```

```

ID TYPE      LOGFILE_TABLE      BADFILE_TABLE      STATUS_TABLE
-----
41 PIPELINE PIPELINE$9$41_LOG PIPELINE$9$41_BAD PIPELINE$9$41_STATUS

```

This query shows ID, TYPE, LOGFILE_TABLE, BADFILE_TABLE if it exists, and the STATUS_TABLE. You can view these tables for additional pipeline load information.

Pipeline Status Table Details

Column	Datatype	Description
ID	NUMBER	Unique number assigned to the pipeline.
NAME	VARCHAR2(4000)	Name of the pipeline.
BYTES	NUMBER	Bytes
CHECKSUM	VARCHAR2(128)	Checksum
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	Last modification time for the pipeline.
STATUS	VARCHAR2(30)	The STATUS value is one of: - COMPLETED: The file operation completed successfully. - FAILED: The file operation failed, a retry may be attempted two times. - PENDING: The file operation has not yet started. - RUNNING: The file operation is currently in progress. - SKIPPED: The file operation skipped.

Column	Datatype	Description
ERROR_CODE	NUMBER	Error code
ERROR_MESSAGE	VARCHAR2(4000)	Error message
START_TIME	TIMESTAMP(6) WITH TIME ZONE	Start time for the pipeline.
END_TIME	TIMESTAMP(6) WITH TIME ZONE	End time for the pipeline.
SID	NUMBER	The session SID and SERIAL# indicate the job session that was running the pipeline load operation.
SERIAL#	NUMBER	The session SID and SERIAL# indicate the job session that was running the pipeline load operation.
ROWS_LOADED	NUMBER	Number of rows loaded.
OPERATION_ID	NUMBER	Reserved for future use.

Pipeline Log File and Bad File Tables

To obtain the log file and bad file names for a load pipeline, query the pipeline's OPERATION_ID. For example:

```
SELECT PIPELINE_NAME, OPERATION_ID FROM USER_CLOUD_PIPELINES
       WHERE PIPELINE_NAME = 'MY_TREE_DATA';
```

```
PIPELINE_NAME OPERATION_ID
-----
MY_TREE_DATA          41
```

Next, query either USER_LOAD_OPERATIONS or DBA_LOAD_OPERATIONS with a WHERE clause predicate on the ID column (using the OPERATION_ID value).

For example:

```
SELECT ID, TYPE, LOGFILE_TABLE, BADFILE_TABLE, STATUS_TABLE FROM
USER_LOAD_OPERATIONS
       WHERE ID = 41;
```

```
ID TYPE      LOGFILE_TABLE      BADFILE_TABLE      STATUS_TABLE
--
41 PIPELINE PIPELINE$9$41_LOG PIPELINE$9$41_BAD PIPELINE$9$41_STATUS
```

This query shows ID, TYPE, LOGFILE_TABLE, BADFILE_TABLE if it exists, and the STATUS_TABLE. You can view these tables for additional pipeline load information.

View the pipeline log file table to see a complete log of the pipeline's load operations.

For example:

```
SELECT * FROM PIPELINE$9$41_LOG;
```

View the pipeline bad file table to see details on input format records with errors. The bad file table show information for the rows reporting errors during loading. Depending on the errors

shown in the log file table and the rows shown in the pipeline's bad file table, you might be able to correct the errors either by modifying the pipeline format attribute options, or by modifying the data in the file you are loading.

For example:

```
SELECT * FROM PIPELINE$9$41_BAD;
```

See [Monitor and Troubleshoot Data Loading](#) for more information.

Bulk Operations for Files in the Cloud

The PL/SQL package `DBMS_CLOUD` offers parallel execution support for bulk file upload, download, copy, and transfer activities, which streamlines the user experience and delivers optimal performance for bulk file operations.

The package `DBMS_CLOUD` supports loading and unloading files into the following cloud services:

- Oracle Cloud Infrastructure Object Storage
- Azure Blob Storage
- Amazon S3
- Google Cloud Storage

See [DBMS_CLOUD Package File URI Formats](#) for more information.

About Bulk File Operations

`DBMS_CLOUD` bulk file operations support downloading and uploading files on Autonomous AI Database, including support for copying, moving, and deleting files across Cloud Object Stores, either within the same Object Store provider or between Object Store providers. The bulk file operations support parallelism and provide optimal performance for uploading, downloading, copying, and moving files. Parallelism for bulk file operations is handled by specifying a priority for an operation. The supported priorities are:

- **HIGH:** Use the Autonomous AI Database instance CPU compute count to determine the number of parallel files processed.
- **MEDIUM:** Use the concurrency limit for Medium service to determine the parallelism.
- **LOW:** Process files in serial order (no concurrency).

Running bulk operations with a higher priority uses more database resources and operations complete faster when parallelism can speed up the operation. A lower priority consumes less database resources and operations take longer to complete when parallelism can speed up the operation. When bulk operations involve small numbers of files that contain little data, specifying higher priority may not change the performance.

To increase parallelism for bulk file operations use **HIGH** priority and increase the number of CPUs assigned to your Autonomous AI Database instance. The maximum supported concurrent file operations is limited to 64 for bulk file operations.

The default priority is **MEDIUM**, which specifies that bulk file operations use the concurrency limit defined for the **MEDIUM** consumer group.

See [DBMS_CLOUD for Bulk File Management](#) for details on using the `format` parameter to specify the priority with bulk file operations.

In the DBMS_CLOUD subprograms discussed below, the `credential_name`, `source_credential_name` or the `target_credential_name` can be NULL for public, Pre-authenticated, or Pre-signed bucket URI.

Bulk Copy Files in Cloud Object Storage

Use the DBMS_CLOUD.BULK_COPY procedure to bulk copy files from one Object Store bucket or folder to another bucket or folder.

1. Create a credential object to access the source location.

The source credential name, as specified with the `source_credential_name` parameter is by default also used as the credential for the target location.

See CREATE_CREDENTIAL for more information.

2. When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, create a credential to access the target location and include the `target_credential_name` parameter.
3. Run DBMS_CLOUD.BULK_COPY procedure to copy files in parallel from one Object Store bucket or folder to another bucket or folder which can be across cloud provider, accounts, and buckets. To run the procedure, you must be logged in as the ADMIN user or have the EXECUTE privilege on DBMS_CLOUD.

```
BEGIN
DBMS_CLOUD.BULK_COPY (
    source_credential_name => 'OCI_CRED',
    source_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/o',
    target_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
    format                 => JSON_OBJECT ('logretention' value 7,
'logprefix' value 'BULKOP')
);
END;
/
```

This example bulk copies files from one Oracle Cloud Infrastructure Object Storage bucket to another.

See BULK_COPY Procedure for more information.

See DBMS_CLOUD Package File Cloud Object Storage URI Formats for more information.

For example, use DBMS_CLOUD.BULK_COPY to copy files from Amazon S3 to Oracle Cloud Infrastructure Object Storage.

```
BEGIN
DBMS_CLOUD.BULK_COPY(
    source_credential_name => 'AWS_CRED',
    source_location_uri    => 'https://bucketname.s3-us-
west-2.amazonaws.com/',
    target_credential_name => 'OCI_CRED',
    target_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
    format                 => JSON_OBJECT ('logretention' value 7,
'logprefix' value 'BULKOP')
```

```
);
END;
/
```

Bulk Move Files Across Cloud Object Storage

Use the `DBMS_CLOUD.BULK_MOVE` procedure to bulk move files from one Cloud Object Storage location to another.

The first step in moving files is copying them to the target location. After the files are successfully copied, they are deleted from the source location.

The files are renamed rather than copied if Object Store allows renaming operations between source and target locations.

1. Create a credential object to access the source location.

The source credential name, as specified with the `source_credential_name` parameter is by default also used as the credential for the target location.

See `CREATE_CREDENTIAL` for more information.

2. When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, create a credential to access the target location and include the `target_credential_name` parameter.
3. Run `DBMS_CLOUD.BULK_MOVE` procedure to bulk move files from one Cloud Object Storage location to another. To run the procedure, you must be logged in as the ADMIN user or have the EXECUTE privilege on `DBMS_CLOUD`.

```
BEGIN
DBMS_CLOUD.BULK_MOVE (
    source_credential_name => 'OCI_CRED',
    source_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/o',
    target_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
    format                 => JSON_OBJECT ('logretention' value 7,
'logprefix' value 'BULKMOVE')
);
END;
/
```

This example bulk moves files from one Oracle Cloud Infrastructure Object Storage location to another.

See `BULK_MOVE` Procedure for more information.

See `DBMS_CLOUD` Package File Cloud Object Storage URI Formats for more information.

For example, use `DBMS_CLOUD.BULK_MOVE` to move files from Amazon S3 to Oracle Cloud Infrastructure Object Storage.

```
BEGIN
DBMS_CLOUD.BULK_MOVE(
    source_credential_name => 'AWS_CRED',
    source_location_uri    => 'https://bucketname.s3-us-
west-2.amazonaws.com/',
```



```

        target_credential_name => 'OCI_CRED',
        target_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
        format                 => JSON_OBJECT ('logretention' value 7,
'logprefix' value 'BULKOP')
    );
END;
/

```

Bulk Download Files from Cloud Object Storage

Use the `DBMS_CLOUD.BULK_DOWNLOAD` procedure to bulk download files from the Cloud Object Store location to the directory object in an Autonomous AI Database.

1. Create a credential to access your Cloud Object Storage.

The credential name, is specified with the `credential_name` parameter.

See `CREATE_CREDENTIAL` for more information.

2. Run `DBMS_CLOUD.BULK_DOWNLOAD` procedure to download files into an Autonomous AI Database directory from Cloud Object Storage. To run the procedure, you must be logged in as the `ADMIN` user or have the `EXECUTE` privilege on `DBMS_CLOUD`.

```

BEGIN
DBMS_CLOUD.BULK_DOWNLOAD (
    credential_name => 'OCI_CRED',
    location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
    directory_name  => 'BULK_TEST',
    format         => JSON_OBJECT ('logretention' value 7, 'logprefix'
value 'BULKOP')
);
END;
/

```

This example bulk downloads files from the Oracle Cloud Infrastructure Object Store location URI to the directory object in an Autonomous AI Database.

Note

To write the files in the target directory object, you must have the `WRITE` privilege on the directory object.

See `BULK_DOWNLOAD` Procedure for more information.

See `DBMS_CLOUD` Package File Cloud Object Storage URI Formats for more information.

Bulk Upload Files to Cloud Object Storage

Use the `DBMS_CLOUD.BULK_UPLOAD` procedure to bulk upload files from a directory object in database to a Cloud Object Store location.

1. Create a credential to access your Cloud Object Storage.

The credential name, is specified with the `credential_name` parameter.

See `CREATE_CREDENTIAL` for more information.

2. Run `DBMS_CLOUD.BULK_UPLOAD` procedure to copy files into Cloud Object Storage from a database directory on your Autonomous AI Database instance. To run the procedure, you must be logged in as the ADMIN user or have the EXECUTE privilege on `DBMS_CLOUD`.

```
BEGIN
DBMS_CLOUD.BULK_UPLOAD (
    credential_name => 'OCI_CRED',
    location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
    directory_name  => 'BULK_TEST',
    format          => JSON_OBJECT ('logretention' value 5, 'logprefix'
value 'BULKUPLOAD')
);
END;
/
```

This example bulk uploads files from a directory object, as specified with the `directory_name` parameter to the Oracle Cloud Infrastructure Object Store location URI.

Note

To read the source files in the directory object, you must have the READ privilege on the directory object.

See `BULK_UPLOAD` Procedure for more information.

Bulk Delete Files from Cloud Object Storage

Use the `DBMS_CLOUD.BULK_DELETE` procedure to bulk delete files from Cloud Object Storage.

1. Create a credential to access your Cloud Object Storage.

The credential name, is specified with the `credential_name` parameter.

The `credential_name` can be NULL for public or Pre-authenticated or Pre-signed bucket URI.

See `CREATE_CREDENTIAL` for more information.

2. Run `DBMS_CLOUD.BULK_DELETE` procedure to delete files from the Cloud Object Store. To run the procedure, you must be logged in as the ADMIN user or have the EXECUTE privilege on `DBMS_CLOUD`.

```
BEGIN
DBMS_CLOUD.BULK_DELETE (
    credential_name => 'OCI_CRED',
    location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
    format          => JSON_OBJECT ('logretention' value 5, 'logprefix'
value 'BULKDEL')
);
```

```
END;  
/
```

This example bulk deletes files from the Oracle Cloud Infrastructure Object Store.

See BULK_DELETE Procedure for more information.

See DBMS_CLOUD Package File Cloud Object Storage URI Formats for more information.

Monitor and Troubleshoot Bulk File Loads

All DBMS_CLOUD data load operations are logged in the dba_load_operations and user_load_operations views.

You can use the following views to monitor and troubleshoot bulk file loads:

- dba_load_operations: shows all load operations.
- user_load_operations: shows the load operations in your schema.

Query these views to see information about ongoing and completed bulk file operations. For example, the following SELECT statement with a WHERE clause predicate on the TYPE shows DOWNLOAD operations:

```
SELECT owner_name, type, status, start_time, update_time, status_table, rows_loaded  
FROM user_load_operations WHERE type = 'DOWNLOAD';
```

OWNER_NAME	TYPE	STATUS	START_TIME	UPDATE_TIME	STATUS_TABLE	ROWS_LOADED

"ADMIN"	DOWNLOAD	COMPLETED	2022-10-17T20:42:19.498Z	2022-10-17T20:42:21.054Z		
DWN\$2_STATUS		4				
"ADMIN"	DOWNLOAD	FAILED	2022-10-17T20:40:54.348Z	2022-10-17T20:40:55.679Z		
DWN\$1_STATUS						

The STATUS_TABLE column shows the name of the table you can query to look at detailed logging information for the bulk download operation. For example:

```
DESCRIBE DWN$2_STATUS
```

Name	Null?	Type

ID	NOT NULL	NUMBER
NAME		VARCHAR2(4000)
BYTES		NUMBER
CHECKSUM		VARCHAR2(128)
LAST_MODIFIED		TIMESTAMP(6) WITH TIME ZONE
STATUS		VARCHAR2(30)
ERROR_CODE		NUMBER
ERROR_MESSAGE		VARCHAR2(4000)
START_TIME		TIMESTAMP(6) WITH TIME ZONE
END_TIME		TIMESTAMP(6) WITH TIME ZONE
SID		NUMBER
SERIAL#		NUMBER
ROWS_LOADED		NUMBER

```

SELECT id, name, bytes, status, error_message, start_time, end_time FROM
DWN$2_STATUS;
ID NAME          BYTES STATUS    ERROR_MESSAGE START_TIME
END_TIME
--
-----
1 trees.txt      58 COMPLETED          2022-10-17T20:42:19.998Z
2022-10-17T20:42:20.421Z
2 trees1.txt     58 COMPLETED          2022-10-17T20:42:20.425Z
2022-10-17T20:42:20.533Z
3 trees2.txt     58 COMPLETED          2022-10-17T20:42:20.535Z
2022-10-17T20:42:20.894Z
4 trees3.txt     58 COMPLETED          2022-10-17T20:42:20.896Z
2022-10-17T20:42:21.043Z

```

The status table shows each file name and its status for the bulk operation.

The relevant error number and message are recorded in the status table if an operation on a specific file fails.

For completed operations, the time needed for each operation can be calculated using the reported START_TIME and END_TIME time.

The file operation STATUS column can have one of the following values:

File Status	Description
COMPLETED	File operation completed successfully.
FAILED	File operation failed, a retry may be attempted for two times.
PENDING	The file operation has not yet started.
RUNNING	File operation is currently in progress.
SKIPPED	File operation skipped.

If any file operation fails after two retry attempts, then the bulk operation is marked as failed and an error is raised. For example:

```
ORA-20003: Operation failed, please query table DOWNLOAD$2_STATUS for error details
```

When you use a DBMS_CLOUD bulk file operation there are format parameter options that control status tables:

- **logretention:** Specifies an integer value that determines the duration in days that the status table is retained. The default value is 2 days.
- **logprefix:** Specifies a string value that determines the bulk operation status table's name prefix.

Each bulk operation has its own default value for the logprefix option:

Procedure	Default Value for logprefix
DBMS_CLOUD.BULK_COPY	COPYOBJ
DBMS_CLOUD.BULK_DELETE	DELETE
DBMS_CLOUD.BULK_DOWNLO	DOWNLOAD
AD	

Procedure	Default Value for logprefix
DBMS_CLOUD.BULK_MOVE	MOVE
DBMS_CLOUD.BULK_UPLOAD	UPLOAD

See DELETE_ALL_OPERATIONS Procedure for information on clearing the user_load_operations table.

Manage Users

Managing Database Users

This section describes administration tasks for managing database users on Autonomous AI Database.

Create Database Users

You can create database users in Autonomous AI Database on Dedicated Exadata Infrastructure by connecting to the database as the ADMIN user using any SQL client tool.

Run the following SQL statements to create a database user:

```
CREATE USER new_user
IDENTIFIED BY password
DEFAULT TABLESPACE tablespace_name;
```

```
GRANT CREATE SESSION TO new_user;
```

Note

IDENTIFIED EXTERNALLY and IDENTIFIED BY VALUES clauses are not supported by Autonomous AI Database on Dedicated Exadata Infrastructure with the CREATE USER SQL statement.

This creates *new_user* with connect privileges. This user can now connect to Autonomous AI Database 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 About Connecting to a Dedicated Autonomous AI Database.

Autonomous AI Database 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:

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

Unlock or Change the ADMIN Database User Password

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

Note

If you have unsuccessfully attempted to authenticate with the ADMIN database user account multiple times, it may be locked out. Changing the ADMIN database user password unlocks the account automatically.

1. Go to the **Details** page of the Autonomous AI Database whose ADMIN user password you want to change.

For instructions, see [View Details of a Dedicated Autonomous AI Database](#).

2. On the **Details** page, select **More Actions** and then select **Administrator Password**.
3. On the **Administrator 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 AI Database come with a predefined database role named DWR0LE. This role provides the privileges necessary for most database users. Here are the privileges it grants a user:

```
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
```

To grant the DWROLE role to a database user, connect to the database as the ADMIN user using any SQL client tool and then enter this SQL statement:

```
GRANT DWROLE TO user;
```

Instead of or in addition to granting DWROLE privileges, you can grant individual privileges to users with the GRANT command.

The DWROLE role does not allocate any tablespace quota to the user. If the user is going to be adding data or other objects, you need to grant the user tablespace quota in one of these ways:

- Grant the user UNLIMITED TABLESPACE privileges:

```
GRANT UNLIMITED TABLESPACE TO user;
```

- Grant the user quota to tablespaces individually; for example:

```
ALTER USER user QUOTA 500M ON data;
```

Note

Granting the UNLIMITED TABLESPACE privilege allows a user to use all the allocated storage space. You cannot selectively revoke tablespace access from a user with the UNLIMITED TABLESPACE privilege. You can grant selective or restricted access only after revoking the privilege.

For further reference:

- See *GRANT* in Oracle Database 19c SQL Language Reference or [Oracle Database 26ai SQL Language Reference](#).
- See *ALTER USER* in Oracle Database 19c SQL Language Reference or [Oracle Database 26ai 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:

```
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 User Profiles with Autonomous AI Database

You can create and alter user profiles in Autonomous AI Database. After you create or alter a profile, you can specify the profile clause with `CREATE USER` or `ALTER USER`. You can also import existing user profiles from another environment with Oracle Data Pump Import.

Note

Autonomous AI Database has restrictions on the profile clause. See [Limitations on the Use of SQL Commands](#) for information on `CREATE PROFILE` and `ALTER PROFILE` restrictions.

To add, modify, or remove a password parameter in a profile, including the `DEFAULT` profile you must have the `ALTER PROFILE` system privilege.

1. To add or alter a profile, as the `ADMIN` user run either `CREATE PROFILE` or `ALTER PROFILE`. For example:

```
CREATE PROFILE new_profile
  LIMIT PASSWORD_REUSE_MAX 10
  PASSWORD_LOCK_TIME 5;
```

If you are not the `ADMIN` user, then you must have `CREATE PROFILE` privilege to run `CREATE PROFILE`. If you run `ALTER PROFILE`, then you must have `ALTER PROFILE` privilege.

2. Use the new or altered profile with a `CREATE USER` or `ALTER USER` command. For example:

```
CREATE USER new_user IDENTIFIED BY password PROFILE new_profile;
GRANT CREATE SESSION TO new_user;
```

This creates `new_user` with profile `new_profile` and with connect privileges. The `new_user` can now connect to the database and run queries. To grant additional privileges to users, see [Manage Database User Privileges](#).

See `CREATE PROFILE` in Oracle Database 19c SQL Language Reference or [Oracle Database 26ai SQL Language Reference](#) for information on using `CREATE PROFILE`.

You can import existing profiles created in other environments using Oracle Data Pump Import (`impdp`). Any existing profile association with database users is preserved after importing into Autonomous AI Database. When a newly created user, created from an Oracle Data Pump import, attempts to login for the first time, the login is handled as follows:

- The password complexity restrictions are the same as the restrictions for any user on Autonomous AI Database.
- If the user's password violates the password complexity requirements, the account is expired with a 30-day grace period. In this case the user is required to change their password before the grace period ends.

Note

Profile assignments for users with profile `ORA_PROTECTED_PROFILE` cannot be modified.

When you create or alter a profile, you can specify a Password Verification Function (PVF) to manage password complexity. See [Manage Password Complexity on Autonomous AI Database](#) for more information.

Manage Password Complexity on Autonomous AI Database

You can create a Password Verify Function (PVF) and associate the PVF with a profile to manage the complexity of user passwords.

Note

The minimum password length for a user specified PVF is 8 characters and must include at least one upper case letter, one lower case letter and one numeric character. The minimum password length for the DEFAULT profile is 12 characters (the DEFAULT profile uses the CLOUD_VERIFY_FUNCTION PVF). The password cannot contain the username.

Oracle recommends using a minimum password length of 12 characters. If you define a profile's PVF, and set the minimum password length to less than 12 characters, then tools such as Oracle Database Security Assessment Tool (DBSAT) and Qualys report this as a database security risk.

For example, to specify a PVF for a profile, use the following command:

```
CREATE PROFILE example_profile LIMIT PASSWORD_VERIFY_FUNCTION  
ADMIN.EXAMPLE_PVF
```

If the profile is created or altered by any user other than the ADMIN user, then you must grant the EXECUTE privilege on the PVF. If you create a PVF and the password check fails, the database reports the ORA-28219 error.

You can specify an Oracle supplied PVF, from one of the following:

- CLOUD_VERIFY_FUNCTION (this is the default password verification function for Autonomous AI Database):
This function checks for the following requirements when users create or modify passwords:
 - The password must be between 12 and 30 characters long and must include at least one uppercase letter, one lowercase letter, and one numeric character.
 - The password cannot contain the username.
 - The password cannot be one of the last four passwords used for the same username.
 - The password cannot contain the double quote (") character.
 - The password must not be the same password that is set less than 24 hours ago.
- ORA12C_STIG_VERIFY_FUNCTION
This function checks for the following requirements when users create or modify passwords:
 - The password has at least 15 characters.
 - The password has at least 1 lower case character and at least 1 upper case character.
 - The password has at least 1 digit.

- The password has at least 1 special character.
- The password differs from the previous password by at least 8 characters.

See *ora12c_stig_verify_function Password Requirements* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for more information.

Note the following restrictions for a Password Verification Function (PVF) that you create and assign to a profile:

- If you specify a user profile, the minimum password length depends on how you define the associated PVF, as follows:
 - If a PVF is defined, then the minimum password length enforced is 8 characters with at least one upper case letter, one lower case letter and one numeric character. The password cannot contain the username.
 - If the PVF is defined as NULL, then the minimum password length enforced is 8 characters with at least one upper case letter, one lower case letter and one numeric character. The password cannot contain the username.
 - If the profile does not have a PVF defined, then the DEFAULT profile's PVF (CLOUD_VERIFY_FUNCTION) is assigned and the minimum password length enforced is 12 characters.
- If you specify a Password Verify Function (PVF) that is more strict than the default CLOUD_VERIFY_FUNCTION, then the new verify function is used.
- A PVF that you create must be created as a DEFINER RIGHTS PL/SQL function. If a INVOKER rights PVF is provided as input to CREATE or ALTER PROFILE, then ORA-28220 error is thrown.
- Any PVF that you create must be created in the ADMIN user schema. If a non-ADMIN user owned PVF is provided as input to CREATE or ALTER PROFILE, then ORA-28220 error is thrown.
- A PVF cannot be altered or dropped by a non-ADMIN user. That is, any user with the CREATE or DROP ANY PROCEDURE privilege is not allowed to alter or drop a PVF.
- If the PVF associated with a profile is dropped, then any attempt to change the password for a user who uses the PVF in their profile throws the error ORA-7443. Users can still login when the PVF associated with their profile is dropped. However, if a user's password is expired and the PVF is dropped, then the user cannot login.

To recover from the ORA-7443 error, the ADMIN user must recreate the dropped PVF and assign it to the profile, or assign an existing PVF to the profile. This allows a user change their password and login.

- The CREATE ANY PROCEDURE system privilege and DROP ANY PROCEDURE system privilege are audited for PVF security. See the PROCEDURES list in *Listings of System and Object Privileges* in Oracle Database 19c SQL Language Reference or [Oracle Database 26ai SQL Language Reference](#) for more information.

See *Managing the Complexity of Passwords* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for more information.

Gradual Database Password Rollover for Applications

An application can change its database passwords without an administrator having to schedule downtime.

To accomplish this, you can associate a profile having a non-zero limit for the `PASSWORD_ROLLOVER_TIME` password profile parameter, with an application schema. This allows the database password of the application user to be altered while allowing the older password to remain valid for the time specified by the `PASSWORD_ROLLOVER_TIME` limit. During the rollover period of time, the application instance can use either the old password or the new password to connect to the database server. When the rollover time expires, only the new password is allowed.

See [Managing Gradual Database Password Rollover for Applications](#) for more information.

Use Oracle Database Vault to Manage Database User Privileges

Autonomous AI Database on Dedicated Exadata Infrastructure supports Oracle Database Vault, which you can use to implement powerful security controls within your database. These unique security controls restrict access to application data by privileged database users, reducing the risk of insider and outside threats and addressing common compliance requirements.

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* in Oracle Database 19c Administrator's Guide or [Oracle Database 26ai Administrator's Guide](#) to gain an understanding of the impact of configuring and enabling Database Vault.

For detailed information on implementing Oracle Database Vault features, see *Introduction to Oracle Database Vault* in Oracle Database 19c Administrator's Guide or [Oracle Database 26ai Administrator's Guide](#).

✓ Tip

To try out the process of setting up Database Vault, you can go through the *Lab 1: Protect Data with Database Vault* in Oracle Autonomous AI Database Dedicated for Security Administrators Workshop.

Configure and Enable Oracle Database Vault

To configure and enable Oracle Database Vault in your Autonomous AI 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 database.
2. Create the Database Vault owner and account manager users; for example:

```
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 database.
5. Enable Database Vault:

```
exec dbms_macadm.enable_dv;
```
6. Restart (that is, stop and then start) your Autonomous AI Database on Dedicated Exadata Infrastructure deployment.

Use Identity and Access Management (IAM) Authentication with Autonomous AI Database

You can configure Autonomous AI Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous AI Database with IAM credentials.

Note

Autonomous AI Database integration with Oracle Cloud Infrastructure (OCI) IAM is supported in commercial tenancies with identity domains as well as the legacy Oracle Cloud Infrastructure IAM, which does not include identity domains. Oracle Cloud Infrastructure IAM with identity domains was introduced with new OCI tenancies created after November 8, 2021. Autonomous AI Database supports users and groups in default and non-default identity domains.

About Identity and Access Management (IAM) Authentication with Autonomous AI Database

Oracle Cloud Infrastructure IAM integration with Autonomous AI Database supports both database password authentication and token-based authentication. See [Authenticating and Authorizing IAM Users for Oracle Autonomous AI Databases](#) for complete details about the architecture for using IAM users on Autonomous AI Database.

IAM Database Password Authentication

Note

Any supported 12c and above database client can be used for IAM database password access to Autonomous AI Database.

An Oracle Cloud Infrastructure IAM database password allows an IAM user to log in to an Autonomous AI Database instance as Oracle Database users typically log in with a user name and password. The user enters their IAM user name and IAM database password. An IAM database password is a different password than the Oracle Cloud Infrastructure Console password. Using an IAM user with the password verifier you can login to Autonomous AI Database with any supported database client.

Identity and Access Management (IAM) SSO Token Based Authentication

There are several ways a database client can obtain an IAM database token:

- A client application or tool can request the database token from IAM for the user and can pass the database token through the client API. Using the API to send the token overrides other settings in the database client. This type of IAM database token usage is supported for the following clients:
 - JDBC-thin on all platforms
 - Oracle Instant Client OCI-C on Linux
 - Oracle Data Provider for .NET (ODP.NET) Core
- If the application or tool does not support requesting an IAM database token through the client API, the IAM user can first use Oracle Cloud Infrastructure command line interface (CLI) to retrieve the IAM database token and save it in a file location. For example, to use SQL*Plus and other applications and tools using this connection method, you first obtain the database token using the Oracle Cloud Infrastructure (OCI) Command Line Interface (CLI). If the database client is configured for IAM database tokens, when a user logs in with the slash login form, the database driver uses the IAM database token that has been saved in a default or specified file location.
- A client application or tool can use an Oracle Cloud Infrastructure IAM instance principal or resource principal to get an IAM database token, and use the IAM database token to authenticate itself to an Autonomous AI Database instance.
- IAM users and OCI applications can request a database token from IAM with several methods, including using an API-key. See [Configuring a Client Connection for SQL*Plus That Uses an IAM Token](#) for an example. See About Authenticating and Authorizing IAM Users for an Oracle Autonomous AI Database for a description of other methods such as using a delegation token within an OCI cloud shell.

If a user enters a username/password to login, then the database driver uses the password verifier method to access the database regardless of the client's database token setting.

Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous AI Database

Before using Identity and Access Management (IAM) authentication on Autonomous AI Database, you must ensure the following:

-
- [Oracle Public Cloud](#)
 - [Exadata Cloud@Customer](#)

Oracle Public Cloud

Use the Networking service to add a service gateway, a route rule, and an egress security rule to the VCN (Virtual Cloud Network) and subnets where your Autonomous AI Database resources reside.

1. Create a service gateway in the VCN (Virtual Cloud Network) where your Autonomous AI Database resources reside by following the instructions in [Task 1: Create the service gateway](#) in *Oracle Cloud Infrastructure Documentation*.

2. After creating the service gateway, add a route rule and an egress security rule to each subnet (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to use Identity and Access Management (IAM) authentication:
 - a. Go to the **Subnet Details** page for the subnet.
 - b. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
 - c. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** All IAD Services In Oracle Services Network
 - **Target Type:** Service Gateway
 - **Target:** The name of the service gateway you just created in the VCNIf such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.
 - d. Return to the **Subnet Details** page for the subnet.
 - e. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
 - f. In the side menu, under **Resources**, click **Egress Rules**.
 - g. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Stateless:** No
 - **Destination:** All IAD Services In Oracle Services Network
 - **IP Protocol:** TCP
 - **Source Port Range:** All
 - **Destination Port Range:** 443
 - h. If such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

Change External Identity Providers on Autonomous AI Database

Describes the steps to change the external identity provider from (IAM) authentication and authorization to another external authentication provider and vice-versa.

Oracle Cloud Infrastructure (IAM) authentication and authorization for users is enabled for newly provisioned Autonomous AI Databases and Autonomous Container Databases, by default. There are other external authentication schemes such as, Centrally Managed Users with Active Directory (CMU-AD), Azure AD, and Kerberos that you can enable for your Autonomous AI Database. However, there can only be one external authentication scheme enabled at any given time.

To enable database users to connect to Autonomous AI Database using another external authentication scheme:

1. Disable IAM integration using the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` procedure:
For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

For additional information, see `DISABLE_EXTERNAL_AUTHENTICATION` Procedure.

2. Follow the steps outlined in the following pages to configure another external authentication scheme of your choice:
 - [Configure CMU with Microsoft Active Directory on Autonomous AI Database](#)
 - [Enable Kerberos Authentication on Autonomous AI Database](#)
 - [Configure the Oracle Autonomous AI Database for Microsoft Azure AD Integration](#)
3. Re-enable IAM users to connect to Autonomous AI Database using Oracle Cloud Infrastructure (IAM) Authentication and Authorization.
As an ADMIN user, run the `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` procedure with the parameters shown below:

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'OCI_IAM'
    force => TRUE
  );
END;
/
```

The above example disables any external authentication provider that is currently enabled and enables Oracle Cloud Infrastructure (IAM) Authentication and Authorization.

In this example:

- `force` is set to `TRUE` to disable the external authentication that is currently enabled.

- type is set to 'OCI_IAM' to enable and use Oracle Cloud Infrastructure (IAM) Authentication and Authorization.

Note

By default the force parameter is FALSE. When another external authentication method is enabled and force is FALSE, DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION reports the following error: ORA-20004: Another external authentication is already enabled.

See ENABLE_EXTERNAL_AUTHENTICATION Procedure for further reference.

4. Verify the value of IDENTITY_PROVIDER_TYPE system parameter as shown below:

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME='identity_provider_type';
NAME                                VALUE
-----
identity_provider_type OCI_IAM
```

Create Identity and Access Management (IAM) Groups and Policies for IAM Users

Describes the steps to write policy statements for an IAM group to enable IAM user access to Oracle Cloud Infrastructure resources, specifically Autonomous AI Database instances.

A policy is a group of statements that specifies who can access particular resources, and how. Access can be granted for the entire tenancy, databases in a compartment, or individual databases. This means you write a policy statement that gives a specific group a specific type of access to a specific type of resource within a specific compartment.

Note

Defining a policy is required to use IAM tokens to access Autonomous AI Database. A policy is not required when using IAM database passwords to access Autonomous AI Database.

To enable Autonomous AI Database to allow IAM users to connect to the database using IAM tokens:

1. Perform Oracle Cloud Infrastructure Identity and Access Management prerequisites by creating a group and adding users to the group.

For example, create the group sales_dbusers.

See [Managing Groups](#) for more information.

2. Write policy statements to enable access to Oracle Cloud Infrastructure resources.
 - a. In the Oracle Cloud Infrastructure console click **Identity & Security**.
 - b. Under **Identity & Security** click **Policies**.
 - c. To a write policy, click **Create Policy**.
 - d. On the Create Policy page, enter a Name and a Description.
 - e. On the Create Policy page, select **Show manual editor**.

Create Policy

Name: sales_test_policy
No spaces. Only letters, numerals, hyphens, periods, or underscores.

Description: Test policy for sales demo

Compartment: example (root)

Policy Builder: Show manual editor ☒

Example: Allow group [group_name] to [verb] [resource-type] in compartment [compartment_name] where [condition]

Create Cancel ☐ Create Another Policy

Terms of Use and Privacy Cookie Preferences Copyright © 2022, Oracle and/or its affiliates. All rights reserved.

f. Use the **Policy Builder** to create a policy.

For example to create a policy to allow users in IAM group DBUsers to access any Autonomous AI Database in their tenancy:

Allow group DBUsers to use autonomous-database-family in tenancy

For example to create a policy that limits members of DBUsers group to access Autonomous AI Databases in compartment testing_compartment only:

```
allow group DBUsers to use autonomous-database-family in compartment
testing_compartment
```

For example to create a policy that limits group access to a single database in a compartment:

```
allow group DBUsers to use autonomous-database-family in compartment
testing_compartment where target.database.id =
'ocid1.autonomoustatabase.oc1.iad.aaaabbbbcccc'
```

Refer to *Creating an IAM Policy to Authorize Users Authenticating with Tokens in Database Security Guide* for more information on IAM policies to access the database.

g. Click **Create**.

See [Managing Policies](#) for more information on policies.

Notes for creating policies for use with IAM users on Autonomous AI Database:

- Policies can allow IAM users to access Autonomous AI Database instances across the entire tenancy, in a compartment, or can limit access to a single Autonomous AI Database instance.

- You can use either instance principal or resource principal to retrieve database tokens to establish a connection from your application to an Autonomous AI Database instance. If you are using an instance principal or resource principal, you must map a dynamic group. Thus, you cannot exclusively map instance and resource principals; you only can map them through a shared mapping and putting the instance or resource instance in an IAM dynamic group.

You can create Dynamic Groups and reference dynamic groups in the policies you create to access Oracle Cloud Infrastructure. See Plan Access Controls and Create Supporting Resources and [Managing Dynamic Groups](#) for details.

Add IAM Users on Autonomous AI Database

To add IAM users to allow access to Autonomous AI Database, map database global users to IAM groups or users with CREATE USER or ALTER USER statements (with IDENTIFIED GLOBALLY AS clause).

The authorization of IAM users to an Autonomous AI Database instance works by mapping IAM global users (schemas) to IAM users (exclusive mapping) or IAM groups (shared schema mapping).

To authorize IAM users on an Autonomous AI Database instance:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required CREATE USER and ALTER USER system privileges that you need for these steps).
2. Create a mapping between the Autonomous AI Database user (schema) with CREATE USER or ALTER USER statements and include the IDENTIFIED GLOBALLY AS clause, specifying the IAM group name.

Use the following syntax to map a global user to an IAM group:

```
CREATE USER global_user IDENTIFIED GLOBALLY AS  
'IAM_GROUP_NAME=IAM_GROUP_NAME';
```

For example, to map an IAM group named db_sales_group to a shared database global user named sales_group:

```
CREATE USER sales_group IDENTIFIED GLOBALLY AS  
'IAM_GROUP_NAME=db_sales_group';
```

This creates a shared global user mapping. The mapping, with global user sales_group, is effective for all users in the IAM group. Thus, anyone in the db_sales_group can log in to the database using their IAM credentials (through the shared mapping of the sales_group global user).

The following example shows how to accomplish this for a non-default domain:

```
CREATE USER shared_sales_schema IDENTIFIED GLOBALLY AS  
'IAM_GROUP_NAME=sales_domain/db_sales_group';
```

3. If you want to create additional global user mappings for other IAM groups or users, follow these steps for each IAM group or user.

Note

Database users that are not IDENTIFIED GLOBALLY can continue to login as before, even when the Autonomous AI Database is enabled for IAM authentication.

To exclusively map a local IAM user to an Oracle Database Global User:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required CREATE USER and ALTER USER system privileges that you need for these steps).
2. Create a mapping between the Autonomous AI Database user (schema) with CREATE USER or ALTER USER statements and include the IDENTIFIED GLOBALLY AS clause, specifying the IAM local IAM user name.

For example, to create a new database global user named peter_fitch and map this user to an existing local IAM user named peterfitch:

```
CREATE USER peter_fitch IDENTIFIED GLOBALLY AS  
'IAM_PRINCIPAL_NAME=peterfitch'
```

The following example shows how to create the user by specifying a non-default domain, sales_domain:

```
CREATE USER peter_fitch2 IDENTIFIED GLOBALLY AS  
'IAM_PRINCIPAL_NAME=sales_domain/peterfitch';
```

Add IAM Roles on Autonomous AI Database

Optionally, create global roles to provide additional database roles and privileges to IAM users when multiple IAM users are mapped to the same shared global user.

The use of global roles is optional when using either an exclusive IAM mapping to user (schema) or shared user mapping on Autonomous AI Database. For example, all privileges and roles can be granted to the shared schema and all IAM users who map to the shared schema would be granted the privileges and roles assigned to the shared schema.

You can use a global role to optionally differentiate users who use the same shared schema. For example, a set of users can all have the same shared schema and the shared schema could have the CREATE SESSION privilege. Then global roles can be used to provide differentiated privileges and roles assigned to different groups of users who all use the same shared schema.

Granting additional roles to IAM users in Autonomous AI Database works by mapping Autonomous AI Database global roles to IAM groups.

To map Autonomous AI Database global roles to IAM groups:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required CREATE USER and ALTER USER system privileges that you need for these steps).
2. Set database authorization for Autonomous AI Database roles with CREATE ROLE or ALTER ROLE statements and include the IDENTIFIED GLOBALLY AS clause, specifying the IAM group name.

Use the following syntax to map a global role to an IAM group:

```
CREATE ROLE global_role IDENTIFIED GLOBALLY AS  
  'IAM_GROUP_NAME=IAM_GROUP_of_WHICH_the_IAM_USER_IS_a_MEMBER';
```

For example, to map an IAM group named ExporterGroup to a shared database global role named export_role:

```
CREATE ROLE export_role IDENTIFIED GLOBALLY AS  
  'IAM_GROUP_NAME=ExporterGroup';
```

The following example shows how to create the role by specifying a non-default domain, sales_domain:

```
CREATE ROLE export_role IDENTIFIED GLOBALLY AS  
  'IAM_GROUP_NAME=sales_domain/ExporterGroup';
```

All members of the ExporterGroup in the sales_domain domain will be authorized with the database global role export_role when they log in to the database.

3. Use GRANT statements to grant the required privileges or other roles to the global role.

```
GRANT CREATE SESSION TO export_role;  
GRANT DWROLE TO export_role;
```

4. If you want an existing database role to be associated with an IAM group, then use ALTER ROLE statement to alter the existing database role to map the role to an IAM group. Use the following syntax to alter an existing database role to map it to an IAM group:

```
ALTER ROLE existing_database_role  
  IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=IAM_Group_Name';
```

If you want to add additional global role mappings for other IAM groups, follow these steps for each IAM group.

Create IAM Database Password for IAM Users

To add an IAM user and allow the IAM user to login to Autonomous AI Database by supplying a username and password, you must create an IAM database password.

See [Working with IAM Database Passwords](#) for more information.

Connect to Autonomous AI Database with Identity and Access Management (IAM) Authentication

After the database ADMIN user maps global users and global roles to the IAM users and IAM groups, users log in to the Autonomous AI Database instance using their Oracle Cloud Infrastructure IAM credentials or access the database through an Oracle Cloud Infrastructure IAM database token.

You can still log in to the Autonomous AI Database using your local database account username and password (non-global database user account).

You can use a database client to access an Autonomous AI Database instance as an Oracle Cloud Infrastructure IAM user. To use a client with Oracle Cloud Infrastructure IAM username and password credentials and a password verifier, the database client must be 12c or newer.

Alternatively, you can use an Oracle Cloud Infrastructure IAM database token to access an Autonomous AI Database instance with supported clients:

- JDBC-Thin with support for IAM Token Authentication is supported with the following:
 - JDBC version 19.13.0.0.1 (or later): See [JDBC and UCP Downloads](#) for JDBC drivers.
 - JDBC version 21.4.0.0.1 (or later): See [JDBC and UCP Downloads](#) for JDBC drivers.

See Support for IAM Token-Based Authentication for more information:

- SQL*Plus and Oracle Instant Client: Supported with SQL*Plus and Instant Client on Linux versions 19.13 or later, and Instant Client on Linux versions 21.4 or later. See Identity and Access Management (IAM) Token-Based Authentication for more information.
- The database client can also be configured to retrieve a database token using the IAM username and IAM database password.
See Client Connections That Use a Token Requested by an IAM User Name and Database Password for more information.
- .NET clients (latest version of Linux or Windows). .NET software components are available as a free download from the following sites:
 - [Oracle Data Access Components - .NET Downloads](#)
 - [NuGet Gallery](#)
 - [Visual Studio Code Marketplace](#)

Note

If your Autonomous AI Database instance is in Restricted Mode, only the users with the RESTRICTED SESSION privilege such as ADMIN can connect to the database.

About Connecting to an Autonomous AI Database on Dedicated Exadata Infrastructure Instance Using IAM

IAM users can connect to the Autonomous AI Database on Dedicated Exadata Infrastructure instance by using either an IAM database password verifier or an IAM token.

Using the IAM database password verifier is similar to the Oracle Database password authentication process. However, instead of the password verifier (encrypted hash of the password) being stored in the Autonomous AI Database on Dedicated Exadata Infrastructure, the verifier is instead stored as part of the Oracle Cloud Infrastructure (OCI) IAM user profile.

The second connection method, the use of an IAM token for the database, is more modern. The use of token-based access is a better fit for Cloud resources such as Autonomous AI Database on Dedicated Exadata Infrastructure. The token is based on the strength that the IAM endpoint can enforce. This can be multi-factor authentication, which is stronger than the use of passwords alone. Another benefit of using tokens is that the password verifier (which is considered sensitive) is never stored or available in memory.

Client Connections That Use an IAM Database Password Verifier

After you have configured the authorization needed for the IAM user, this user can log in using existing client application, such as SQL*Plus or SQLcl without additional configuration.

The IAM user enters the IAM user name and IAM database password (not the Oracle Cloud Infrastructure (OCI) console password) using any currently supported database client. The only constraint is that the database client version be either Oracle Database release 12.1.0.2 or later to use Oracle Database 12c passwords. The database client must be able to use the 12C password verifier. Using the 11G verifier encryption is not supported with IAM. No special client or tool configuration is needed for the IAM user to connect to the Autonomous AI Database on Dedicated Exadata Infrastructure instance.

Client Connections That Use a Token

For IAM token access to the Autonomous AI Database on Dedicated Exadata Infrastructure, the client application or tool requests a database token from IAM for the IAM user.

The client application will pass the database token directly to the database client through the database client API.

If the application or tool has not been updated to request an IAM token, then the IAM user can use Oracle Cloud Infrastructure (OCI) command line interface (CLI) to request and store the database token. You can request a database access token (db-token) using the following credentials:

- Security tokens (with IAM authentication), delegation tokens (in the OCI cloud shell) and API-keys, which are credentials that represent the IAM user to enable the authentication
- Instance principal tokens, which enable instances to be authorized actors (or principals) to perform actions on service resources after authenticating
- Resource principal token, which is a credential that enables the application to authenticate itself to other Oracle Cloud Infrastructure services

When the IAM users logs into the client with a slash / login and the OCI_IAM parameter is configured (sqlnet.ora, tnsnames.ora, or as part of a connect string), then the database client retrieves the database token from a file. If the IAM user submits a user name and password, the connection will use the IAM database verifier access described for client connections that use IAM database password verifiers. The instructions in this guide show how to use the OCI CLI as a helper for the database token. If the application or tool has been updated to work with IAM, then follow the instructions for the application or tool. Some common use cases include the following: SQL*Plus on-premises, SQLcl on-premises, SQL*Plus in Cloud Shell, or applications that use SEP wallets.

The following topics explain how to:

- Configure a client connection for SQL*Plus to use an IAM database password.
- Configure a client connection for SQL*Plus that uses an IAM token.
- Access the database through an Oracle Cloud Infrastructure IAM database token using an instance principal.

Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password

You can configure SQL*Plus to use an IAM database password.

- As the IAM user, log in to the Autonomous AI Database instance by using the following syntax:

```
CONNECT user_name@db_connect_string  
Enter password: password
```

In this specification, *user_name* is the IAM user name. There is a limit of 128 bytes for the combined *domain_name/user_name*.

The following example shows how IAM user *peter_fitch* can log in to an Autonomous AI Database instance.

```
sqlplus /nolog  
connect peter_fitch@db_connect_string  
Enter password: password
```

Some special characters will require double quotation marks around *user_name* and *password*. For example:

```
"peter_fitch@example.com"@db_connect_string  
"IAM database password"
```

Configuring a Client Connection for SQL*Plus That Uses an IAM Token

You can configure a client connection for SQL*Plus that uses an IAM token.

1. Ensure you have an IAM user account.
2. Check with an IAM administrator and an Oracle Database administrator to ensure you have a policy allowing you to access the database in the compartment or your tenancy and that you are mapped to a global schema in the database.
3. If your application or tool does not support direct IAM integration, then download, install, and configure the OCI CLI. (See [OCI Command Line Interface Quickstart](#).) Set up an API key as part of the OCI CLI configuration and select default values.
 - a. Set up the API key access for the IAM user.
 - b. Retrieve the db-token. For example:
 - Retrieving a db-token with an API-key using the Oracle Cloud Infrastructure (OCI) command-line interface:

```
oci iam db-token get
```

- Retrieving a db-token with a security (or session) token:

```
oci iam db-token get --auth security_token
```

If the security token has expired, a window will appear so the user can log in to OCI again. This generates the security token for the user. OCI CLI will use this refreshed token to get the db-token.

- Retrieving a db-token with a delegation token: When you log in to the cloud shell, the delegation token is automatically generated and placed in the /etc directory. To get this token, execute the following command in the cloud shell:

```
oci iam db-token get
```

- Retrieving an instance token by using the OCI command-line interface:

```
oci iam db-token get --auth instance_principal
```

- c. The database client can also be configured to retrieve a database token using the IAM username and IAM database password.

See [Client Connections That Use a Token Requested by an IAM User Name and Database Password](#) for more information.

See [Required Keys and OCIDs](#) for more information.

4. Ensure that you are using the latest release updates for the Oracle Database client releases 19c and 21c.

This configuration only works with the Oracle Database client release 19c or 21c.

5. Follow the existing process to download the wallet from the Autonomous AI Database and then follow the directions for configuring it for use with SQL*Plus.

- a. Confirm that DN matching is enabled by looking for SSL_SERVER_DN_MATCH=ON in `sqlnet.ora`.
- b. Configure the database client to use the IAM token by adding `TOKEN_AUTH=OCI_TOKEN` to the `sqlnet.ora` file. Because you will be using the default locations for the database token file, you do not need to include the token location.

The `TOKEN_AUTH` and `TOKEN_LOCATION` values in the `tnsnames.ora` connect strings take precedence over the `sqlnet.ora` settings for that connection. For example, for the connect string, assuming that the token is in the default location (`~/.oci/db-token` for Linux):

```
(description=
  (retry_count=20)(retry_delay=3)
  (address=(protocol=tcps)(port=1522)
  (host=example.us-phoenix-1.oraclecloud.com))

(connect_data=(service_name=aaabbbccc_exampledb_high.example.oraclecloud.com))
  (security=(ssl_server_cert_dn="CN=example.uscom-east-1.oraclecloud.com,
    OU=Oracle BMCS US, O=Example Corporation,
    L=Redwood City, ST=California, C=US")
  (TOKEN_AUTH=OCI_TOKEN)))
```

After the connect string is updated with the `TOKEN_AUTH` parameter, the IAM user can log in to the Autonomous AI Database instance by running the following command to start SQL*Plus. You can include the connect descriptor itself or use the name of the descriptor from the `tnsnames.ora` file.

```
connect /@exampledb_high
```


Or:

```
connect /@(description=
  (retry_count=20)(retry_delay=3)
  (address=(protocol=tcps)(port=1522)
  (host=example.us-phoenix-1.oraclecloud.com))

(connect_data=(service_name=aaabbbccc_exampledb_high.example.oraclecloud.com))
  (security=(ssl_server_cert_dn="CN=example.uscom-east-1.oraclecloud.com,
    OU=Oracle BMCS US, O=Example Corporation,
    L=Redwood City, ST=California, C=US")
  (TOKEN_AUTH=OCI_TOKEN)))
```

The database client is already configured to get a db-token because TOKEN_AUTH has already been set, either through the sqlnet.ora file or in a connect string. The database client gets the db-token and signs it using the private key and then sends the token to the Autonomous AI Database. If an IAM user name and IAM database password are specified instead of slash /, then the database client will connect using the password instead of using the db-token.

Use Instance Principal to Access Autonomous AI Database with Identity and Access Management (IAM) Authentication

After the ADMIN user enables Oracle Cloud Infrastructure IAM on Autonomous AI Database, an application can access the database through an Oracle Cloud Infrastructure IAM database token using an instance principal.

See [Accessing the Oracle Cloud Infrastructure API Using Instance Principals](#) for more information.

Configure Proxy Authentication

Proxy authentication allows an IAM user to proxy to a database schema for tasks such as application maintenance.

Proxy authentication is typically used to authenticate the real user and then authorize them to use a database schema with the schema privileges and roles in order to manage an application. Alternatives such as sharing the application schema password are considered insecure and unable to audit which actual user performed an action.

A use case can be in an environment in which a named IAM user who is an application database administrator can authenticate by using their credentials and then proxy to a database schema user (for example, hrapp). This authentication enables the IAM administrator to use the hrapp privileges and roles as user hrapp in order to perform application maintenance, yet still use their IAM credentials for authentication. An application database administrator can sign in to the database and then proxy to an application schema to manage this schema.

You can configure proxy authentication for both the password authentication and token authentication methods.

Configuring Proxy Authentication for the IAM User

To configure proxy authentication for an IAM user, the IAM user must already have a mapping to a global schema (exclusive or shared mapping). A separate database schema for the IAM user to proxy to must also be available.

After you ensure that you have this type of user, alter the database user to allow the IAM user to proxy to it.

1. Log in to the Autonomous AI Database instance as a user who has the ALTER USER system privileges.
2. Grant permission for the IAM user to proxy to the local database user account. An IAM user cannot be referenced in the command so the proxy must be created between the database global user (mapped to the IAM user) and the target database user. In the following example, hrapp is the database schema to proxy to, and peterfitch_schema is the database global user exclusively mapped to user peterfitch.

```
ALTER USER hrapp GRANT CONNECT THROUGH peterfitch_schema;
```

At this stage, the IAM user can log in to the database instance using the proxy. For example:

To connect using a password verifier:

```
CONNECT peterfitch[hrapp]@connect_string
Enter password: password
```

To connect using a token:

```
CONNECT [hrapp]/@connect_string
```

Validating the IAM User Proxy Authentication

You can validate the IAM user proxy configuration for both password and token authentication methods.

1. Log in to the Autonomous AI Database instance as a user who has the CREATE USER and ALTER USER system privileges.
2. Connect at the IAM user and execute the SHOW USER and SELECT SYS_CONTEXT commands. For example, suppose you want to check the proxy authentication of the IAM user peterfitch when they proxy to database user hrapp. You will need to connect to the database using the different types of authentication methods shown here, but the output of the commands that you execute will be the same for all types.

- For password authentication:

```
CONNECT peterfitch[hrapp]/password\!@connect_string
SHOW USER;
--The output should be USER is "HRAPP"
SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;
--The output should be "PASSWORD_GLOBAL"
SELECT SYS_CONTEXT('USERENV','PROXY_USER') FROM DUAL;
--The output should be "PETERFITCH_SCHEMA"
SELECT SYS_CONTEXT('USERENV','CURRENT_USER') FROM DUAL;
--The output should be "HRAPP"
```

- For token authentication:

```
CONNECT [hrapp]/@connect_string
SHOW USER;
--The output should be USER is "HRAPP "
SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;
```

```
--The output should be "TOKEN_GLOBAL"  
SELECT SYS_CONTEXT('USERENV','PROXY_USER') FROM DUAL;  
--The output should be "PETERFITCH_SCHEMA"  
SELECT SYS_CONTEXT('USERENV','CURRENT_USER') FROM DUAL;  
--The output should be "HRAPP"
```

Use Database Link with IAM Authenticated Users

You can use a database link to connect from one Autonomous AI Database instance to another as an Oracle Cloud Infrastructure IAM user.

You can use either connected user or fixed user database link to connect to an Autonomous AI Database as an Oracle Cloud Infrastructure IAM user.

Note

Current user database link is not supported for connecting to an Autonomous AI Database as an Oracle Cloud Infrastructure IAM user.

- **Connected User Database Link:** For a connected user database link, an IAM user must be mapped to a schema in both the source and target databases connected by a database link. You can use a database password verifier or an IAM database token to use a connected user database link.
- **Fixed User Database Link:** A fixed user database link can be created using a database user or an IAM user. When using an IAM user as a fixed user database link, the IAM user must have a schema mapping in the target database. The IAM user for a database link can be configured with a password verifier only.

Disable Identity and Access Management (IAM) Authentication on Autonomous AI Database

Describes the steps to disable IAM external authentication user access for Autonomous AI Database.

You can disable IAM user access on your Autonomous AI Database instance using the ALTER SYSTEM command as shown below:

```
ALTER SYSTEM RESET IDENTITY_PROVIDER_TYPE SCOPE=BOTH;
```

If you also want to update access to IAM from the resource, that is, Autonomous Container Database or Autonomous AI Database, you may need to remove or modify the IAM group and the policies you set up to allow access to IAM from those resources.

Notes for Using Autonomous AI Database Tools with Identity and Access Management (IAM) Authentication

Provides notes for using Autonomous AI Database tools with IAM authentication enabled.

- Oracle APEX is not supported for IAM users with Autonomous AI Database. See Access Oracle Application Express Administration Services for information on using regular database users with Autonomous AI Database.

- Database Actions is not supported for IAM users with Autonomous AI Database. See [Enabling User Access to Database Actions](#) for information on using regular database users with Autonomous AI Database.

About Kerberos Authentication

You can configure Autonomous AI Database on Dedicated Exadata Infrastructure to use Kerberos network authentication protocol to authenticate database users. Kerberos is a strong network authentication protocol. It uses secret-key cryptography to enable strong authentication by providing user-to-server authentication.

- Autonomous AI Database on Dedicated Exadata Infrastructure support for Kerberos provides the benefits of single sign-on and centralized authentication of Oracle users. Kerberos is a trusted third-party authentication system that relies on shared secrets. It presumes that the third party is secure, and provides single sign-on capabilities, centralized password storage, database link authentication, and enhanced PC security. It does this through a Kerberos authentication server.
- The Kerberos system revolves around the concept of a ticket. A ticket is a set of electronic information that identifies a user or a service. A ticket identifies you and your network access privileges.
- In Kerberos-based authentication, you transparently send a request for a ticket to a Key Distribution Center (KDC). The Key Distribution Center authenticates you and grants you a ticket to access the database.

Components of the Kerberos Authentication System

Provides an overview of the Kerberos authentication system.

- A realm establishes an authentication administrative domain. Each realm has its own Kerberos database which contains the users and services for that particular administrative domain.
- Tickets are issued by the Key Distribution Center (KDC). Clients present tickets to the Database Server to demonstrate the authenticity of their identity. Each ticket has expiration and a renewal time.
- Keytabs stores long-term keys for one or more principals. A keytab file is generated by invoking the tool `kadmin.local` (for MIT Key Distribution Center) or `ktpass` (for Active Directory Key Distribution Center).
- Principals are the entries in the Key Distribution Center database. Each user, host or service is given a principal. A principal is a unique identity to which the Key Distribution Center can assign tickets.
- Kerberos support in Autonomous AI Database uses these values for various components that make up a service principal's name:

Component of Service Principal	Value in Autonomous AI Database
kinstance	You can obtain this value from the attribute <code>PUBLIC_DOMAIN_NAME</code> in the <code>CLOUD_IDENTITY</code> column of <code>V\$PDBS</code>. This value is different from the Fully Qualified Domain Name (FQDN) for a database on a private endpoint. Use the following query to obtain the kinstance: <pre>SELECT json_value(cloud_identity, '\$.PUBLIC_DOMAIN_NAME') "KINSTANCE" FROM v\$pdb;</pre> Note This will be the value of the host parameter found in the TNS connect string.
kservice	On Autonomous AI Database you have two options for the kservice value: - Use the database GUID: If you do not provide a Kerberos service name with <code>DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION</code>, the default service name used is the GUID of the Autonomous AI Database instance. In this case, when you create the keytab file, use the GUID as the value for the service name. Because in the default case the Keytab file uses a service name that is a GUID, which is instance specific, when you use the default service name you must generate different Keytab files for each Autonomous AI Database instance. Use the following command to obtain the GUID (case is significant): <pre>SELECT GUID FROM v\$pdb;</pre> - Use a custom name: Set the service name when you want to use the same Keytab files on multiple Autonomous AI Database instances. When you use a custom name you do not need to create and upload different Keytab files for each Autonomous AI Database instance. When you use a custom name you must specify the <code>params.kerberos_service_name</code> parameter with <code>DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION</code>. Specifying this parameter is not required when you use the GUID for the service name. After Kerberos is enabled on your Autonomous AI Database instance, use the following query to view the Kerberos service name: <pre>SELECT SYS_CONTEXT('USERENV', 'KERBEROS_SERVICE_NAME') FROM DUAL;</pre>
REALM	Any realm supported by your KDC. REALM must always be in uppercase.

To enable Kerberos authentication for your Autonomous AI Database, you must keep your Kerberos configuration files (`krb.conf`) and service key table file (`v5srvtab`) ready. For more information on these files and steps to obtain them, please see [Configuring Kerberos Authentication](#).

Notes About Kerberos Authentication on Autonomous AI Database

Before proceeding with Kerberos authentication on Autonomous AI Database on Dedicated Exadata Infrastructure, review the following notes:

- If you enable Kerberos authentication for your Autonomous AI Database, you can still use password-based database authentication for your database.
- Only one external authentication method can be used for your Autonomous AI Database at any time. That is, you can only have Oracle Cloud Infrastructure (IAM), Centrally Managed User with Active Directory (CMU-AD), Azure AD, or Kerberos authentication schemes enabled at any time.

Note

The only exception is that Kerberos authentication can be configured on top of CMU-AD to provide CMU-AD Kerberos authentication for Microsoft Active Directory users.

- Kerberos authentication is not supported for the following tools:
 - Oracle Database API for MongoDB
 - Oracle REST Data Services
 - Oracle Machine Learning
 - APEX
 - Oracle Graph Studio
 - Oracle Database Actions
- You can enable Kerberos authentication to authenticate the ADMIN user. You can use the Reset Password functionality on the Oracle Cloud Infrastructure (OCI) Console to reset the ADMIN user's password and regain access if a corrupted keytab file causes ADMIN user's authentication to fail.
- Kerberos authentication is supported with TCP protocol for Autonomous AI Database versions 19.27 or later.
- Kerberos authentication is not supported with DB_LINKs and databases with Autonomous Data Guard.

Enable Kerberos Authentication on Autonomous AI Database

Follow the below-listed steps to enable Kerberos authentication for Autonomous AI Database on Dedicated Exadata Infrastructure:

- As there can only be one external authentication scheme enabled for an Autonomous AI Database at any given time, run the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` procedure to disable any external authentication scheme that is already enabled for your database.

To run the procedure, you must be logged in as ADMIN user or have the EXECUTE privilege on `DBMS_CLOUD_ADMIN`.

```
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

- Obtain the Kerberos configuration files: `krb.conf` and the service key table file `v5srvtab`. For more information on these files and steps required to obtain them, see *Configuring Kerberos Authentication* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#).

- Copy the Kerberos configuration files `krb.conf` and `v5srvtab` to a bucket in your Object Store.

Note

Oracle recommends that you store the Kerberos configuration files in a private bucket in your Object Store.

If you are using Oracle Cloud Infrastructure Object Store, see [Putting Data into Object Storage](#) for details on uploading files.

- To enable Kerberos as the **external authentication** for your Autonomous AI Database, run the `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` procedure and pass in a location URI with the `params JSON` argument. You must place the configuration files `krb.conf` and `v5srvtab` in the Object Storage location specified in the `location_uri` parameter.

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type      => 'KERBEROS',
    params    => JSON_OBJECT('location_uri' value 'https://
objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o',
                                'credential_name' value
'my_credential_name')
  );
END;
/
```

Note

To provide **CMU-AD Kerberos authentication** for Microsoft Active Directory users, you can enable Kerberos authentication on top of CMU-AD by setting type to CMU in the above example.

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

The `credential_name` you use in this step is the credentials for the Object Store.

If the `location_uri` is a pre-authenticated URL then supplying a `credential_name` is not required.

This creates a directory object named `KERBEROS_DIR` in your database and uses the credential to download the Kerberos configuration files from the Object Store location to the directory object.

You can use the `kerberos_service_name` parameter to specify the Kerberos service name. For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
```

```

type      => 'KERBEROS',
params    => JSON_OBJECT(
  'location_uri'          value 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
  'credential_name'       value 'my_credential_name'
  'kerberos_service_name' value 'oracle' ));
END;
/

```

- After you enable Kerberos authentication, remove the configuration `krb.conf` and `v5srvtab` from Object Store. You can use local Object Store methods to remove these files or use `DBMS_CLOUD.DELETE_OBJECT` to delete the files from Object Store.

See [Navigate to Oracle Cloud Infrastructure Object Storage and Create Bucket](#) for more information on Object Storage.

See `ENABLE_EXTERNAL_AUTHENTICATION` Procedure for more information.

Disable Kerberos Authentication on Autonomous AI Database

Before enabling any other external authentication scheme on your database, you must disable the Kerberos authentication running the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` procedure.

To run this procedure, you must be logged in as ADMIN user or have the EXECUTE privilege on `DBMS_CLOUD_ADMIN`.

```

BEGIN
  DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/

```

See `DISABLE_EXTERNAL_AUTHENTICATION` Procedure for more information.

Note

To provide CMU-AD Kerberos authentication for the Microsoft Active Directory users, you must continue to configure CMU-AD authentication without disabling Kerberos authentication.

Authenticate and Authorize Microsoft Azure AD Users for Autonomous AI Database

An Oracle Autonomous AI Database on Dedicated Exadata Infrastructure instance can be configured for Microsoft Azure AD users to connect using Azure OAuth2 access tokens.

About Integrating Oracle Autonomous AI Database on Dedicated Exadata Infrastructure with Microsoft Azure AD

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure and Microsoft Azure AD can be configured to allow users and applications to connect to the database using their Azure AD credentials.

Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to access the database. This is done with an Azure AD OAuth2 access token that the user or application first requests from Azure AD. This OAuth2 access token contains the user identity and database access information and is then sent to the database. Refer to the Microsoft article [Passwordless authentication options for Azure Active Directory](#) for information about configuring multi-factor and passwordless authentication.

You can perform this integration in the following Oracle Database environments:

- On-premises Oracle Database release 19.18 and later
- Oracle Autonomous AI Database on Shared Exadata Infrastructure
- Oracle Autonomous AI Database on Dedicated Exadata Infrastructure
- Oracle Base Database Service
- Oracle Exadata Cloud Service (Oracle ExaCS)

The instructions for configuring Azure AD use the term "Oracle Database" to encompass these environments.

This type of integration enables the Azure AD user to access an Oracle Autonomous AI Database on Dedicated Exadata Infrastructure instance. Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to get an Azure AD OAuth2 access token to send to the database.

The Azure AD administrator creates and registers Oracle Autonomous AI Database on Dedicated Exadata Infrastructure with Azure AD. Within Azure AD, this is called an app registration, which is short for application registration. This is the digital information that Azure AD must know about the software that is using Azure AD. The Azure AD administrator also creates application (app) roles for the database app registration in Azure AD. App roles connect Azure users, groups, and applications to database schemas and roles. The Azure AD administrator assigns Azure AD users, groups, or applications to the app roles. These app roles are mapped to a database global schema or a global role or to both a schema and a role. An Azure AD user, group, or application that is assigned to an app role will be mapped to a database global schema, global role, or to both a schema and a role. An Oracle global schema can also be mapped exclusively to an Azure AD user. An Azure AD guest user (non-organization user) or an Azure AD service principal (application) can only be mapped to a database global schema through an Azure AD app role. An Oracle global role can only be mapped from an Azure app role and cannot be mapped from an Azure user.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure tools including Oracle APEX, Database Actions, Oracle Graph Studio, and Oracle Database API for MongoDB are not compatible with using Azure AD tokens to connect with the database.

Tools and applications that are updated to support Azure AD tokens can authenticate users directly with Azure AD and pass the database access token to the Oracle Autonomous AI Database on Dedicated Exadata Infrastructure instance. You can configure existing database tools such as SQL*Plus to use an Azure AD token from a file location. In these cases, Azure AD tokens can be retrieved using tools like Microsoft PowerShell or Azure CLI and put into a file location. An Azure AD OAuth2 database access tokens are issued with an expiration time. The Oracle Database client driver will ensure that the token is in a valid format and that it has not expired before passing it to the database. The token is scoped for the database, which means that there is information in the token about the database where the token will be used. The app roles the Azure AD principal was assigned to in the database Azure AD app registration are included as part of the access token. The directory location for the Azure AD token should only have enough permission for the user to write the token file to the location and the database client to retrieve these files (for example, just read and write by the user). Because the token allows access to the database, it should be protected within the file system.

Azure AD users can request a token from Azure AD using a number of methods to open an Azure login window to enter their Azure AD credentials.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure accepts tokens representing the following Azure AD principals:

- Azure AD user, who is registered user in the Azure AD tenancy
- Guest user, who is registered as a guest user in the Azure AD tenancy
- Service, which is the registered application connecting to the database as itself with the client credential flow (connection pool use case)

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure supports the following Azure AD authentication flows:

- Authorization code, most commonly used for human users (not applications) to authenticate to Azure AD in a client environment with a browser
- Client credentials, which are for database applications that connect as themselves (and not the end-user)
- On-Behalf-Of (OBO), where an application requests an access token on behalf of a logged-in user to send to the database
- Resource owner password credential (ROPC), which is not recommended for production use, but can be used in test environments where a pop-up browser user authentication would be difficult to incorporate. ROPC needs the Azure AD user name and password credential to be part of the token request call.

Architecture of the Microsoft Azure AD Integration with an Autonomous AI Database

Microsoft Azure Active Directory tokens follow the OAuth2 standard with extensions. Using an Azure AD token to access an Oracle database is similar to using OCI IAM tokens. See *Architecture of the Microsoft Azure AD Integration with an Oracle Database* in *Security Guide* for a detailed explanation.

Azure AD Users Mapping to the Autonomous AI Database

Microsoft Azure users must be mapped to an Autonomous AI Database schema and have the necessary privileges (through roles) before being able to authenticate to the Autonomous AI Database instance. See *Azure AD Users Mapping to the Oracle Database* in *Security Guide* for information about the different ways of mapping users, groups, and application in Microsoft Azure.

Use Cases for Connecting to an Autonomous AI Database Using Azure AD

Oracle Database supports three types of use cases for connecting to an Autonomous AI Database instance using Microsoft Azure Active Directory. See *Use Cases for Connecting to an Oracle Database Using Azure AD* for more details.

General Process of Authenticating Microsoft Azure AD Identities with Oracle Autonomous AI Database on Dedicated Exadata Infrastructure

The Oracle Database administrator and the Microsoft Azure AD administrator play roles to allow Azure AD users to connect to the database using Azure AD OAuth2 access tokens.

The general process is as follows:

1. The Oracle Database administrator ensures that the Oracle Database environment meets the requirements for the Microsoft Azure AD integration. See Oracle Database Requirements for the Microsoft Azure AD Integration.
2. The Azure AD administrator creates an Azure AD app registration for the database and the Oracle Database administrator enables the database to use Azure AD tokens for database access.
As part of the app registration process, the Azure AD administrator creates Azure app roles to be used for the mappings between the Azure users, groups, and applications to the Oracle Database schemas and roles.
3. The Oracle Database administrator creates and maps global schemas to either an Azure AD user (exclusive schema mapping) or to an Azure app role (shared schema mapping). The Azure AD user or application must be mapped to one schema.
4. Optionally, the Oracle administrator creates and maps global Oracle Database roles to Azure app roles.
5. The Azure AD end user who wants to connect with the Oracle Autonomous AI Database on Dedicated Exadata Infrastructure instance registers the client application as an Azure AD client (similar to how the Oracle database is registered).
The Azure AD client will have a client identification and a client secret, unless the application client is public. If the application client is public, then only the application client identification is necessary.
6. The Azure AD end user (who can be a database administrator) connects using an utility such as PowerShell or the Azure command-line interface to retrieve the OAuth2 database access token and store it in a local file directory. An application can also request an Azure AD OAuth2 access token directly from Azure AD and pass it through a database client API. Refer to the following Oracle Database client documentation for information about passing Azure AD OAuth2 tokens:
 - JDBC-thin clients: *Oracle AI Database JDBC Developer's Guide*
 - Oracle Call Interface (OCI): *Oracle Call Interface Developer's Guide*
 - Oracle Data Provider for .NET (ODP): *Oracle Data Provider for .NET Developer's Guide* Connecting to Oracle Database
7. Once connected to the Oracle Autonomous AI Database on Dedicated Exadata Infrastructure instance, the Azure AD end user performs database operations as needed.

Enable Azure AD Authentication on Autonomous AI Database

An Azure AD administrator and an Autonomous AI Database administrator perform steps to configure Azure AD authentication on Autonomous AI Database.

Prerequisites

The Microsoft Azure AD integration with the Oracle Autonomous AI Database on Dedicated Exadata Infrastructure requires:

1. The Autonomous AI Database to be *version 19.18* or higher.
2. Connectivity to the database on TLS port 2484. Non TLS connections are not supported.
3. Outbound network connectivity to Azure AD so that the database can request the Azure AD public key.
4. The Autonomous AI Database to be registered with Azure AD.
5. For Exadata Cloud@Customer deployments, the HTTP Proxy settings in your environment must allow the database to use Azure AD.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

Procedure

Implement the following tasks to configure your Autonomous AI Database for Microsoft Azure AD integration.

1. **Register the Autonomous AI Database instance with a Microsoft Azure AD tenancy:**
A user with Azure AD administrator privileges uses Microsoft Azure AD to register the Oracle Database instance with the Microsoft Azure AD tenancy. Refer to [Register the Oracle Autonomous AI Database Instance with a Microsoft Azure AD Tenancy](#) in *Security Guide*.
2. **Enable Microsoft Azure AD v2 Access Tokens:** If your organization uses the Microsoft Azure AD v2 access token (instead of v1 tokens), then you may need to make additional changes in Azure AD to support the `upn: claim` in your token. See [Enabling Microsoft Azure AD v2 Access Tokens and Checking the Azure AD Access Token Version](#) in *Security Guide*.
3. **Manage app roles in Microsoft Azure AD:** In Azure AD, you can create and manage app roles that will be assigned to Azure AD users and groups and also be mapped to Oracle Database global schemas and roles. Refer to [Manage App Roles in Microsoft Azure AD](#) in *Security Guide*.
4. **Configure outbound connectivity to Microsoft Azure AD using a NAT gateway:**
Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 - a. Go to the **Subnet Details** page for the subnet.
 - b. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
 - c. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** NAT Gateway
 - **Target:** The name of the NAT gateway you just created in the VCN

If such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.

- d. Return to the **Subnet Details** page for the subnet.
- e. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
- f. In the side menu, under **Resources**, click **Egress Rules**.
- g. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** All

If such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

5. Test the Accessibility of the Entra ID Endpoint

Ensure that your Oracle Database instance can access the Entra ID endpoint by following the steps outlined in Testing the Accessibility of the Azure Endpoint in *Security Guide*.

If the database cannot connect with the Microsoft Entra ID endpoint, even after you set the ACL policy, check the above listed prerequisites to confirm that you have configured the network properly per the prerequisites. If the problem persists, please review your networking to ensure the database instance can connect to the MS Entra ID endpoint.

6. Configure Azure AD as an external identity provider for Autonomous AI Database:

By default, Autonomous AI Databases and Autonomous Container Databases are configured to connect users with Oracle Cloud Infrastructure (IAM) authentication and authorization. An application DBA can also change this to another external authentication scheme, such as Centrally Managed Users with Active Directory (CMU-AD) or Kerberos. However, an Autonomous AI Database can enable only one external authentication scheme at a time.

To enable Azure AD as an external identity provider on an Autonomous AI Database instance:

- a. Log in to the Autonomous AI Database instance as a user who has the EXECUTE privilege on the DBMS_CLOUD_ADMIN PL/SQL package. The ADMIN user has this privilege.
- b. As there can only be one external authentication scheme enabled for an Autonomous AI Database at any given time, run the DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION procedure to disable any external authentication scheme that is already enabled for your database. To run the procedure, you must be logged in as ADMIN user or have the EXECUTE privilege on DBMS_CLOUD_ADMIN.

```
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

- c. Run the `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` procedure with the Azure AD required parameters.

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'AZURE_AD',
    params => JSON_OBJECT('tenant_id' VALUE 'tenant_id',
                          'application_id' VALUE 'application_id',
                          'application_id_uri' VALUE
'application_id_uri'),
    force => TRUE
  );
END;
```

In this procedure the Azure AD parameters are:

- `type`: Specifies the external authentication provider. For Azure AD, as shown, use `'AZURE_AD'`.
- `params`: Values for the required Azure AD parameters are available from the Azure portal on the app registration Overview pane for Azure Active Directory. The required params for Azure AD are:
 - `tenant_id`: Tenant ID of the Azure Account. Tenant ID specifies the Autonomous AI Database instance's Azure AD application registration.
 - `application_id`: Azure Application ID created in Azure AD to assign roles/schema mappings for external authentication in the Autonomous AI Database instance.
 - `application_id_uri`: Unique URI assigned to the Azure Application. This is the identifier for the Autonomous AI Database instance. The name must be domain qualified (this supports cross tenancy resource access).
The maximum length for this parameter is 256 characters.
- `force`: Set this parameter to `TRUE` if another `EXTERNAL AUTHENTICATION` method is configured for the Autonomous AI Database instance and you want to disable it.

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'AZURE_AD',
    params => JSON_OBJECT('tenant_id' VALUE '29981886-6fb3-44e3-82',
                          'application_id' VALUE '11aa1a11-aaa',
                          'application_id_uri' VALUE 'https://
example.com/111aa1aa'),
    force => TRUE
  );
END;
```

This sets the `IDENTITY_PROVIDER_TYPE` system parameter.

For example, you can use the following to verify `IDENTITY_PROVIDER_TYPE`:

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME='identity_provider_type';
```

NAME	VALUE

```
-----  
identity_provider_type AZURE_AD
```

See `ENABLE_EXTERNAL_AUTHENTICATION` Procedure for more information.

Map Oracle Database Schemas and Roles

Azure AD users will be mapped to one database schema and optionally to one or more database roles.

You have the following options for mapping Oracle database schemas and roles to Microsoft Azure AD users:

- **Exclusively Map an Oracle Database Schema to a Microsoft Azure AD User:** Refer to Exclusively Mapping an Oracle Database Schema to a Microsoft Azure AD User in *Security Guide*.
- **Map a Shared Oracle Schema to an App Role:** Refer to Mapping a Shared Oracle Schema to an App Role in *Security Guide*.
- **Map an Oracle Database Global Role to an App Role:** Refer to Map an Oracle Database Global Role to an App Role in *Security Guide*.

Configure Client Connections to Azure ADs

There are numerous ways that you can configure a client to connect with an Oracle Autonomous AI Database on Dedicated Exadata Infrastructure instance using Azure AD tokens.

You should choose the client connection method that works best with your environment. This guide provides examples of connecting SQL*Plus with different methods of getting an Azure AD OAuth2 access token. All Oracle Database release 19c clients can accept a token that is passed as a file. The JDBC-thin, Instant Client, and ODP.net drivers also accept the token through the database client API from an application. Oracle Database tools such as SQL*Plus cannot retrieve the tokens directly, so tools such as PowerShell or Azure CLI must be used to retrieve the Azure AD OAuth2 access token. To retrieve an Azure AD token, the client must be registered through the Azure AD app registration process. Registering the client is similar to registering the Oracle Autonomous AI Database on Dedicated Exadata Infrastructure server with Azure AD using app registration. Both the database and client must be registered with Azure AD.

The database must be registered so the client can get permission to get an access token for the database. The client must be registered so that Azure AD can recognize a trusted client is asking for an access token.

Note

On the client, you must set the `TOKEN_AUTH` and `TOKEN_LOCATION` parameters in the `sqlnet.ora` file to retrieve the Azure AD database access token from a location and use it when the `/` slash login is used. Exact details are discussed in [Configure SQL*Plus for Azure AD Access Tokens](#).

See the following Microsoft Azure articles for more information about connecting clients to Azure AD:

- [Quickstart: Configure a client application to access a web API](#)

- [Choose the right Azure command-line tool](#)
- [Get Azure AD tokens by using the Microsoft Authentication Library](#)
- [Install the Azure CLI on Linux](#)

Operational Flow for SQL*Plus Client Connection in PowerShell to Autonomous AI Database

The connection between the Azure user, Azure AD, and the Autonomous AI Database instance relies on the passing of the OAuth2 token throughout these components.

Refer to Operational Flow for SQL*Plus Client Connection in PowerShell to Oracle Database in *Security Guide* for an example that shows the use of the Resource Owner Password Credential (ROPC) flow with a public client.

Registering a Client with Azure AD Application Registration

This type of registration is similar to registering Autonomous AI Database with Azure AD app registration. Refer to the following sections for more details:

- Register the database client with Azure as either confidential or public depending on your use case: Confidential and Public Client Registration
- Register a Database Client App with Azure AD: Registering a Database Client App with Azure AD

Examples of Retrieving Azure AD OAuth2 Tokens

For examples showing the different ways that you can retrieve Azure AD OAuth2 tokens, refer to Examples of Retrieving Azure AD OAuth2 Tokens in *Security Guide*.

Configure SQL*Plus for Azure AD Access Tokens

You must configure SQL*Plus to retrieve the Azure AD database access token from a location and use it when the / slash login is used. Refer to Configuring SQL*Plus for Azure AD Access Tokens in *Security Guide* for detailed instructions.

Troubleshooting Microsoft Entra ID Connections

You can use trace files to diagnose problems with Microsoft Entra ID connections. You also can easily remedy ORA-12599 and ORA-03114 errors.

- You can use trace files to troubleshoot the Oracle Database integration with Microsoft Azure AD. See Trace Files for Troubleshooting Oracle Database Client Connections with Azure AD from *Database Security Guide* for guidance.
- The ORA-12599: TNS: cryptographic checksum mismatch and ORA-03114: not connected to ORACLE errors indicate that the database to which you are trying to connect is protected by native network encryption.

When tokens are being used to access an Oracle database, a Transport Layer Security (TLS) connection must be established, not network native encryption. To remedy these errors, ensure that TLS is properly configured for your database. You should test the configuration with a local database user name and password and check the following SYSCONTEXT USERENV parameters:

- NETWORK_PROTOCOL
- TLS_VERSION

- You can check the version of the Microsoft Azure AD access token that your site uses by using the JSON Web Tokens web site. See *Checking the Azure AD Access Token Version* in *Database Security Guide* for guidance.

Use Microsoft Active Directory with Autonomous AI Database on Dedicated Exadata Infrastructure

You can configure Autonomous AI Database on Dedicated Exadata Infrastructure to authenticate and authorize Microsoft Active Directory users. This configuration allows Active Directory users to access an Autonomous AI Database using their Active Directory credentials.

Note

See *Use Azure Active Directory (Azure AD) with Autonomous AI Database* for information on using Azure Active Directory with Autonomous AI Database. The CMU option supports Microsoft Active Directory servers but does not support the Azure Active Directory service.

The integration of Autonomous AI Database with Centrally Managed Users (CMU) provides integration with Microsoft Active Directory. CMU with Active Directory works by mapping Oracle database global users and global roles to Microsoft Active Directory users and groups.

Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous AI Database

The following are required prerequisites to configure the connection from Autonomous AI Database to Active Directory:

- You must have Microsoft Active Directory installed and configured. See [AD DS Getting Started](#) for more information.
- You must create an Oracle service directory user in Active Directory. See *Step 1: Create an Oracle Service Directory User Account on Microsoft Active Directory and Grant Permissions* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for information on the Oracle service directory user account.
- An Active Directory system administrator must have installed Oracle password filter on the Active Directory servers, and set up Active Directory groups with Active Directory users to meet your requirements.

Only password authentication is supported with CMU for Autonomous AI Database, so you must use the included utility, `opwdintg.exe`, to install the Oracle password filter on Active Directory, extend the schema, and create three new `ORA_VFR` groups for three types of password verifier generation. See *Step 2: For Password Authentication, Install the Password Filter and Extend the Microsoft Active Directory Schema* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for information on installing the Oracle password filter.

- The Active Directory servers must be accessible from Autonomous AI Database through the public internet and the port 636 of the Active Directory servers must be open to Autonomous AI Database in Oracle Cloud Infrastructure, so that Autonomous AI Database can have secured LDAP access over TLS/SSL to the Active Directory servers through the internet.

You can also extend your on-premise Active Directory to Oracle Cloud Infrastructure, where you can set up Read Only Domain Controllers (RODCs) for the on-premise Active Directory. Then you can use these RODCs in Oracle Cloud Infrastructure to authenticate and authorize the on-premise Active Directory users for access to Autonomous AI Databases.

See [Extend Active Directory integration in Hybrid Cloud](#) for more information.

- You need the CMU configuration database wallet, `cwallet.sso` and the CMU configuration file `dsi.ora` to configure CMU for your Autonomous AI Database:
 - If you have configured CMU for an on-premise database, you can obtain these configuration files from your on-premise database server.
 - If you have not configured CMU for an on-premise database, you need to create these files. Then you upload the configuration files to the cloud to configure CMU on your Autonomous AI Database instance. You can validate the wallet and the `dsi.ora` by configuring CMU for an on-premise database and verifying that an Active Directory user can successfully log on to the on-premise database with these configuration files. Then you upload these configuration files to the cloud in order to configure CMU for your Autonomous AI Database.

For details on the wallet file for CMU, see:

- *Step 6: Create the Wallet for a Secure Connection* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#)
- *Step 8: Verify the Oracle Wallet* in Oracle Database 19c Security Guide and [Oracle Database 26ai Security Guide](#).

For details on the `dsi.ora` file for CMU, see *Creating the dsi.ora File* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#).

For details on configuring Active Directory for CMU and troubleshooting CMU for on-premise databases, see [How To Configure Centrally Managed Users For Database Release 18c or Later Releases \(Doc ID 2462012.1\)](#).

Configure CMU with Microsoft Active Directory on Autonomous AI Database

To configure Autonomous AI Database for CMU to connect to Active Directory servers:

1. Connect to the Autonomous AI Database as the ADMIN user.
2. Verify if another external authentication scheme is enabled on your database, and disable it.

Note

You can continue with CMU-AD configuration on top of Kerberos to provide CMU-AD Kerberos authentication for Microsoft Active Directory users.

3. Upload the CMU configuration files, including the database wallet file, `cwallet.sso` and the CMU configuration file, `dsi.ora` to your Object Store. This step depends on the Object Store you use.

The `dsi.ora` configuration file contains the information to find the Active Directory servers.

If you are using Oracle Cloud Infrastructure Object Store, see [Putting Data into Object Storage](#) for details on uploading files.

4. On your Autonomous AI Database, create a new directory object or choose an existing directory object. This is the directory where you store the wallet and the configuration file for connecting to Active Directory:

For example:

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

Use the following SQL statement to query the file system directory path of the directory object:

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

For example:

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

```
DIRECTORY_PATH
```

```
-----
```

```
-
```

```
/file_system_directory_path_example/cmu_wallet
```

Note

The directory object name in the query must be upper case as its case was not preserved when the directory object was created.

If you want to preserve case for the directory object name then you need to include its name in double quotes. For example:

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

5. Use DBMS_CLOUD.GET_OBJECT to copy the CMU configuration files, the database wallet `cwallet.sso` and `dsi.ora`, from your Object Store to the directory that you created or chose in the step 4 above.

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

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

```
END;  
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

See GET_OBJECT Procedure for more information.

Use the following SQL statement to query the files copied to the directory.

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('directory_object_name');
```

For example:

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('CMU_WALLET_DIR');
```

Note that the directory object name in this query must be upper case as its case was not preserved when the directory object was created.

6. Enable CMU-AD in your Autonomous AI Database using the DBMS_CLOUD_ADMIN package.

Note

Replace the directory names in the below example with those chosen for your environment. Ensure you are logged in as the ADMIN user before running this command.

```
BEGIN  
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(  
    type      => 'CMU',  
    params    => JSON_OBJECT('directory_name' value 'CMU_WALLET_DIR')  
  );  
END;  
/
```

7. To maintain security, remove the CMU configuration files including the database wallet `cwallet.sso` and the CMU configuration file `dsi.ora` from Object Store. You can use local Object Store methods to remove these files or use `DBMS_CLOUD.DELETE_OBJECT` to delete the files from Object Store.

See DELETE_OBJECT Procedure for more information on `DBMS_CLOUD.DELETE_OBJECT`.

Note

See Disable Active Directory Access on Autonomous AI Database for instructions to disable the access from Autonomous AI Database to Active Directory.

For more information, see *Configuring Centrally Managed Users with Microsoft Active Directory* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#).

Configure CMU with Microsoft Active Directory on Exadata Cloud@Customer

APPLIES TO:  Exadata Cloud@Customer only

To configure Autonomous AI Database on Exadata Cloud@Customer for CMU to connect to Active Directory servers, without using Oracle Object Store service:

1. Connect to the Autonomous AI Database as the ADMIN user.
2. Verify if another external authentication scheme is enabled on your database, and disable it using the following SQL command.

```
BEGIN
  DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

3. CMU-AD needs your Active Directory connection wallet `cwallet.sso` and the `dsi.ora` files on a local file system on your Autonomous Exadata VM Cluster (AVMC). You can achieve this by hosting these files in Oracle Object Store service on Oracle Cloud Infrastructure and then copying them locally using the `DBMS_CLOUD` package. You can find this process with detailed steps and examples in [Configure CMU with Microsoft Active Directory on Autonomous AI Database](#).
4. If hosting `cwallet.sso` and `dsi.ora` in cloud storage is not possible, you may use an Network File System (NFS) share in your data center to host these files, and then move them to a database directory under the Database File System (DBFS). To do so, you must first attach a locally available NFS share to the Autonomous AI Database directory object as demonstrated below:
 - a. Create a Database directory in your Autonomous AI Database instance using the following SQL command from your SQL client:

```
create or replace directory TMPFSSDIR as 'tmpfssdir';
```
 - b. Mount your NFS share to this directory using the `DBMS_CLOUD_ADMIN` package available in Autonomous AI Database.

 Tip

You may need to work with your network or storage administrator to make an NFS share available.

```
BEGIN
  DBMS_CLOUD_ADMIN.attach_file_system(
    file_system_name => <some_name_you_assign>,
    file_system_location => <your_nfs_fs_path>,
    directory_name => <tmpfssdir_created_above>,
    description => 'Any_desc_you_like_to_give'
  );
END
```

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.attach_file_system(
    file_system_name => 'AD-FSS',
    file_system_location => acme.com:/nfs/mount1',
    directory_name => 'TMPFSSDIR',
    description => 'nfs to host AD files'
  );
END;
```

5. To avoid a dependency on the NFS share for the `cwallet.sso` and `dsi.ora` files to be available to CMU, move them to a local file system folder using a database directory mapping. Since Autonomous AI Database restricts access to the local file system, create a copy procedure using `utl_file` as demonstrated below:

- a. Create a Database directory in your Autonomous AI Database instance using the following SQL command from your SQL client:

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

- b. Check the directory path of the above created directory using the following SQL command:

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

Note

The directory object name must be in upper case in the query, as its case was not preserved while creating the directory object.

- c. Copy `dsi.ora` and `cwallet.sso` from the NFS directory to the local CMU Wallet directory using the `UTL_FILE` utility.

For example:

Create a stored procedure called `copyfile` as shown below:

```
CREATE OR REPLACE PROCEDURE copyfile(
  in_loc_dir IN VARCHAR2,
  in_filename IN VARCHAR2,
  out_loc_dir IN VARCHAR2,
  out_filename IN VARCHAR2
)
IS
  in_file UTL_FILE.file_type;
  out_file UTL_FILE.file_type;
  buffer_size CONSTANT INTEGER := 32767;
  buffer RAW (32767);
  buffer_length INTEGER;
BEGIN
  in_file := UTL_FILE.fopen (in_loc_dir, in_filename, 'rb',
    buffer_size);
```

```

    out_file := UTL_FILE.fopen (out_loc_dir, out_filename, 'wb',
buffer_size);
    UTL_FILE.get_raw (in_file, buffer, buffer_size);
    buffer_length := UTL_RAW.LENGTH (buffer);

    WHILE buffer_length > 0
    LOOP
        UTL_FILE.put_raw (out_file, buffer, TRUE);

        IF buffer_length = buffer_size
        THEN
            UTL_FILE.get_raw (in_file, buffer, buffer_size);
            buffer_length := UTL_RAW.LENGTH (buffer);
        ELSE
            buffer_length := 0;
        END IF;
    END LOOP;

    UTL_FILE.fclose (in_file);
    UTL_FILE.fclose (out_file);
EXCEPTION
    WHEN NO_DATA_FOUND
    THEN
        UTL_FILE.fclose (in_file);
        UTL_FILE.fclose (out_file);
END;
/

```

Compile the copyfile stored procedure. Once successfully compiled, run the copyfile procedure once each to copy dsi.ora and cwallet.sso from the NFS directory to the local CMU Wallet directory, as demonstrated below:

```
EXEC copyfile('TMPFSSDIR','dsi.ora','CMU_WALLET_DIR','dsi.ora');
```

```
EXEC copyfile('TMPFSSDIR','cwallet.sso','CMU_WALLET_DIR','cwallet.sso');
```

- d. Run the following SQL query to validate if the files are copied to the local CMU Wallet directory successfully.

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('CMU_WALLET_DIR');
```

6. Using the following command, detach the NFS share since you do not need it for CMU-AD after the files are copied to the local directory.

```
exec DBMS_CLOUD_ADMIN.detach_file_system(file_system_name =>
<FILE_SYSTEM_NAME>);
```

7. Enable CMU-AD in your Autonomous AI Database using the DBMS_CLOUD_ADMIN package.

Note

Replace the directory names in the below example with those chosen for your environment. Ensure you are logged in as the ADMIN user before running this command.

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type      => 'CMU',
    params    => JSON_OBJECT('directory_name' value 'CMU_WALLET_DIR')
  );
END;
/
```

8. Validate by querying the property value of the database property CMU_WALLET as shown below.

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

For Example:

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

PROPERTY_VALUE
-----
CMU_WALLET_DIR
```

You have now configured CMU-AD to use external authentication via Microsoft Active Directory with your Autonomous AI Database on Exadata Cloud@Customer.

Add Microsoft Active Directory Roles on Autonomous AI Database

To add Active Directory roles, map the database global roles to Active Directory groups with CREATE ROLE or ALTER ROLE statements (and include the IDENTIFIED GLOBALLY AS clause).

To add global roles for Active Directory groups on Autonomous AI Database:

1. Log in as the ADMIN user to the database that is configured to use Active Directory (the ADMIN user has the CREATE ROLE and ALTER ROLE system privileges that you need for these steps).
2. Set the database authorization for Autonomous AI Database roles with CREATE ROLE or ALTER ROLE statement. Include the IDENTIFIED GLOBALLY AS clause and specify the DN of an Active Directory group.

Use the following syntax to map a directory user group to a database global role:

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


For example:

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS  
      'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com';
```

In this example all members of the `widget_sales_group` are authorized with the database role `widget_sales_role` when they log in to the database.

3. Use GRANT statements to grant the required privileges or other roles to the global role.

For example:

```
GRANT CREATE SESSION TO WIDGET_SALES_ROLE;  
GRANT DWROLE TO WIDGET_SALES_ROLE;
```

DWROLE is a predefined role that has common privileges defined. See *Manage Database User Privileges* for information on setting common privileges for Autonomous AI Database users.

4. If you want to make an existing database role to be associated with an Active Directory group, then use ALTER ROLE statement to alter the existing database role to map the role to an Active Directory group.

Use the following syntax to alter an existing database role to map it to an Active Directory group:

```
ALTER ROLE existing_database_role  
      IDENTIFIED GLOBALLY AS  
      'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
```

5. If you want to create additional global role mappings for other Active Directory groups, follow these steps for each Active Directory group.

For more information on configuring roles with Microsoft Active Directory, see *Configuring Authorization for Centrally Managed Users* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#).

Add Microsoft Active Directory Users on Autonomous AI Database

To add Active Directory users to access an Autonomous AI Database, map database global users to Active Directory groups or users with CREATE USER or ALTER USER statements (with IDENTIFIED GLOBALLY AS clause).

The integration of Autonomous AI Database with Active Directory works by mapping Microsoft Active Directory users and groups directly to Oracle database global users and global roles.

To add global users for Active Directory groups or users on Autonomous AI Database:

1. Log in as the ADMIN user to the database that is configured to use Active Directory (the ADMIN user has the required CREATE USER and ALTER USER system privileges that you need for these steps).
2. Set database authorization for Autonomous AI Database users with CREATE USER or ALTER USER statements and include the IDENTIFIED GLOBALLY AS clause, specifying the DN of an Active Directory user or group.

Use the following syntax to map a directory user to a database global user:

```
CREATE USER global_user IDENTIFIED GLOBALLY AS 'DN_of_an_AD_USER' ;
```

Use the following syntax to map a directory group to a database global user:

```
CREATE USER global_user IDENTIFIED GLOBALLY AS  
'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER' ;
```

For example, to map a directory group named `widget_sales_group` in the `sales` organization unit of the `production.example.com` domain to a shared database global user named `WIDGET_SALES`:

```
CREATE USER widget_sales IDENTIFIED GLOBALLY AS  
'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com' ;
```

This creates a shared global user mapping. The mapping, with global user `widget_sales`, is effective for all users in the Active Directory group. Thus, anyone in the `widget_sales_group` can log in to the database using their Active Directory credentials (through the shared mapping of the `widget_sales` global user).

3. If you want Active Directory users to use an existing database user, own its schema, and own its existing data, then use `ALTER USER` to alter an existing database user to map the user to an Active Directory group or user.

- Use the following syntax to alter an existing database user to map it to an Active Directory user:

```
ALTER USER existing_database_user IDENTIFIED GLOBALLY AS  
'DN_of_an_AD_USER' ;
```

- Use the following syntax to alter an existing database user to map it to an Active Directory group:

```
ALTER USER existing_database_user  
IDENTIFIED GLOBALLY AS  
'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER' ;
```

4. If you want to create additional global user mappings for other Active Directory groups or users, follow these steps for each Active Directory group or user.

For more information on configuring roles with Microsoft Active Directory, see *Configuring Authorization for Centrally Managed Users* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#).

Connect to Autonomous AI Database with Active Directory User Credentials

After the ADMIN user completes the CMU Active Directory configuration steps and creates global roles and global users, users log in to the Autonomous AI Database using their Active Directory username and password.

Note

Do not log in using a Global User name. Global User names do not have a password and connecting with a Global User name will not be successful. You must have a global user mapping in your Autonomous AI Database in order to log in to the database. You cannot log in to the database with only global role mappings.

- To log in to the Autonomous AI Database using an Active Directory username and password, connect as follows:

```
CONNECT "AD_DOMAIN\AD_USERNAME" /  
AD_USER_PASSWORD@TNS_ALIAS_OF_THE_AUTONOMOUS_DATABASE;
```

For example:

```
CONNECT "production\pfitch"/password@adbname_medium;
```

You need to include double quotes when the Active Directory domain is included along with the username, as with this example: "production\pfitch".

In this example, the Active Directory username is pfitch in domain production. The Active Directory user is a member of widget_sales_group group which is identified by its DN 'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com'.

After configuring CMU with Active Directory on Autonomous AI Database and setting up Active Directory authorization, with global roles and global users, you can connect to your Autonomous AI Database using any of the connection methods described in About Connecting to a Dedicated Autonomous AI Database . When you connect, if you want to use an Active Directory user then use Active Directory user credentials. For example, provide a username in this form, "AD_DOMAIN\AD_USERNAME" (double quotes must be included), and use your AD_USER_PASSWORD for the password.

Verify Active Directory User Connection Information with Autonomous AI Database

When users log in to the Autonomous AI Database using their Active Directory username and password, you can verify and audit the user activity.

For example, when the user pfitch logs in:

```
CONNECT "production\pfitch"/password@exampleadb_medium;
```

The Active Directory user's log on username (samAccountName) is pfitch and widget_sales_group is the Active Directory Group name, and widget_sales is the Autonomous AI Database global user.

After pfitch logs in to the database, the command `SHOW USER` shows the global user name:

```
SHOW USER;

USER is "WIDGET_SALES"
```

The following command shows the DN (Distinguished Name) of the Active Directory user:

```
SELECT SYS_CONTEXT('USERENV', 'ENTERPRISE_IDENTITY') FROM DUAL;
```

For example you can verify this centrally managed user's enterprise identity:

```
SQL> SELECT SYS_CONTEXT('USERENV', 'ENTERPRISE_IDENTITY') FROM DUAL;

SYS_CONTEXT('USERENV', 'ENTERPRISE_IDENTITY')
-----
cn=Peter Fitch,ou=sales,dc=production,dc=examplecorp,dc=com
```

The following command shows the "AD_DOMAIN\AD_USERNAME":

```
SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATED_IDENTITY') FROM DUAL;
```

For example, the Active Directory authenticated user identity is captured and audited when the user logs on to the database:

```
SQL> SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATED_IDENTITY') FROM DUAL;

SYS_CONTEXT('USERENV', 'AUTHENTICATED_IDENTITY')
-----
production\pfitch
```

See *Verifying the Centrally Managed User Logon Information* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for more information.

Remove Active Directory Users and Roles on Autonomous AI Database

To remove Active Directory users and roles from Autonomous AI Databases, use standard database commands. This does not remove the related Active Directory users or groups that were mapped from the dropped database users or roles.

To remove users or roles from Autonomous AI Database:

1. Log in to the database that is configured to use Active Directory as a user who has been granted the `DROP USER` or `DROP ROLE` system privilege.
2. Drop the global users or the global roles that are mapped to Active Directory groups or users with `DROP USER` or `DROP ROLE` statement.

See *Remove Database Users* for more information.

Disable Active Directory Access on Autonomous AI Database

Describes the steps to remove the CMU configuration from your Autonomous AI Database (and disable the LDAP access from your Autonomous AI Database to Active Directory).

After you configure your Autonomous AI Database instance to access CMU Active Directory, you can disable the access as follows:

1. Connect to the Autonomous AI Database as the ADMIN user.
2. Use the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` to disable CMU authentication.

Note

To run this procedure you must be logged in as ADMIN user or have the EXECUTE privilege on `DBMS_CLOUD_ADMIN`.

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

This disables CMU authentication on your Autonomous AI Database instance.

See `DISABLE_EXTERNAL_AUTHENTICATION` Procedure for more information.

Limitations with Microsoft Active Directory on Autonomous AI Database

The following limitations apply to CMU with Active Directory on Autonomous AI Database:

- Only "password authentication" and Kerberos is supported for CMU with Autonomous AI Database. When you are using CMU authentication with Autonomous AI Database, other authentication methods such as Azure AD, OCI IAM, and PKI are not supported.
- Oracle Application Express and Database Actions is not supported for Active Directory users with Autonomous AI Database.

Develop

Developing Applications with Autonomous AI Database

Developing applications that use a database is quite similar to developing applications that use an Oracle database within your own data center. Because the database is running on Oracle Autonomous AI Database and is running on dedicated infrastructure available only within your tenancy, there are several differences that simplify the task of developing a continuously available, high-performing, high-security application:

- **Automatic database administration.** Routine database administration tasks such as patching and taking backups are performed automatically, so you can concentrate on developing your application. See *Service Maintenance* for more information.

- **Automatic indexing and performance tuning.** You need to spend less time defining and honing your database's structure, thanks to these and other features of Oracle Autonomous AI Database. See [Automatic Indexing](#) for more information.
- **Preconfigured continuous availability and high performance.** When you connect to the database through an Oracle client, and you code using connection pools, you can automatically take advantage of the continuous availability and high performance features already configured on the database side of your connection. See [Code for Continuous Availability and Connection Pools](#) for more details.
- **Predefined, workload-specific database services.** With five pairs of predefined database services, you can connect securely to the database using the database connection service that matches the type of database operations you need to perform. See [Predefined Database Service Names for Autonomous AI Database](#) for further reference.

Autonomous AI Database for Developers is a free Autonomous AI Database that developers can use to build and test new applications. It supports all the developer features and tools provided by Oracle Autonomous AI Database, except for Autonomous Data Guard, Database In-Memory, long-term backups, and auto-scaling. Notable developer features included with a developer database are backup and restore, patching, cloning, APEX, ORDS, Database Actions, Performance Hub, Metrics, Notifications, and APIs.

Development Workflow

Create a Database

Usually, DBAs create Autonomous AI Databases and then provide developers with the connection information and credential zip necessary to connect to a database. These DBAs have Oracle Cloud user accounts that grant them authorization to create and manage Autonomous AI Databases on dedicated infrastructure.

If you are such a DBA in addition to being a developer, you can use one of the following Oracle Cloud Infrastructure interfaces to create an Autonomous AI Database:

- **The Oracle Cloud console.** For instructions, see [Create a Dedicated Autonomous AI Database](#).
- **The Oracle Cloud Infrastructure REST API.** For information, see the [CreateAutonomousDatabase](#) endpoint in *Oracle Cloud Infrastructure API Documentation*.
- **The Oracle Cloud Infrastructure CLI.** For information, see the [oci db autonomous-database create](#) command in *OCI CLI Command Reference*.
- **An Oracle Cloud Infrastructure SDK or Plug-in.** For information, see [Software Development Kits](#) and [DevOps Tools and Plug-ins](#) in *Oracle Cloud Infrastructure API Documentation*.

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* in Oracle Database 19c SQL Language Reference or [Oracle Database 26ai SQL Language Reference](#) for more information.

Note

- `CREATE DIRECTORY` creates the database directory object in the Autonomous AI 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 database's data storage allocation.

Manage Directories

Autonomous AI Database on Dedicated Exadata Infrastructure provides you the ability to create directories in your 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.

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* in Oracle Database 19c SQL Language Reference or [Oracle Database 26ai SQL Language Reference](#) for more information.

① Note

- 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 AI 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.

Configure a Development System to Access the Database

Before you can begin developing an application, you must ensure that the development system you are using has network access to the database you want to connect to.

The network visibility of an Autonomous AI Database is determined by the networking configuration of the dedicated infrastructure hosting it.

This networking configuration, which is designed and implemented by the infrastructure's Fleet Administrator and your Oracle Cloud tenancy's network administrator, can range from very open access to very restricted access, depending on the security needs of the databases that are to be created on the infrastructure. To learn more about this networking configuration, see [Plan Access Controls and Create Supporting Resources](#).

Here are the two commonest networking configurations, together with instructions for configuring a development system to have network access to a database created in the configuration.

- **Private Oracle Cloud network.** Databases do not have public IP addresses and are not network-visible to the public internet.

In this case, you need to create an Oracle Cloud Compute instance and install software on it to make it a development system that meets your needs. Depending on the specifics of the network configuration, you may create this instance in the same VCN (virtual cloud network) and subnet as the database, or in the same VCN but a different subnet, or in a different VCN and subnet that have been network-peered to the database's VCN.

To simplify creating such an instance, Oracle provides a Marketplace image that already contains many development tools and utilities. For instructions on using this image to create a development system, see [Compute VM as a Development System](#).

- **Private Oracle Cloud network with a gateway to a VPN.** Databases do not have public IP addresses and are not network-visible to the public internet; however, the private cloud network in which they reside has a gateway to a VPN.

In this case, your existing development system has network access to a database after it is connected to the VPN.

Here are two less common networking configurations, together with instructions for configuring a development system to have network access to a database created in the configuration.

- **Public internet with a gateway.** Databases are network-visible to a specific list of IP addresses on the public internet.

In this case, you need get the public IP address of your existing development system added to the internet gateway that limits public access. The DBA who created the database you want to connect to should be able to provide you instructions to do this.

- **Public internet.** Databases are network-visible to the public internet.

In this case, your existing development system has network access to databases, provided that your system has internet access .

Compute VM as a Development System

Oracle provides an Oracle Cloud Infrastructure Marketplace image that you can use to create an Oracle Cloud Compute instance to serve as your application development system. This image is preconfigured to contain many development tools and utilities.

To use this Marketplace image to create a development system, follow the instructions in the *Lab 8: Configuring a Development System* in Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop.

To go through this lab, note that you will need:

- The name of the compartment, VCN and subnet to use when creating the Oracle Cloud Compute instance. Your autonomous ai database fleet administrator or tenancy administrator should be able to provide you this information.
- An ssh key to use when creating the Oracle Cloud Compute instance.
- An Oracle Cloud user account that has access rights to manage Compute instances and App Catalog listings in the given compartment, to use the given VCN and subnet, and to view information about the database you will be connecting to.

Define or Load the Database Structure

After an Autonomous AI Database has been created, the first step toward developing applications that use the database is to define the database structure.

In general, you define the structure of the database in one of two ways:

- **Manually "from scratch"**

You use a database tool like Oracle SQL Developer or SQL*Plus to connect to the database and define the database's tables, views, triggers, types, sequences and so on. For information about various tools you can use and how to connect them to an Autonomous AI Database, see Oracle Tools for Database Access.

- **By loading structure (and data) from an existing Oracle Database**

You use Oracle Data Pump to export from an existing Oracle Database, upload the export files to cloud storage, and then import the uploaded files into the Autonomous AI Database. For detailed instructions, see Load Data Using Oracle Data Pump.

Regardless of which technique you use, you should review Oracle Database Features in Dedicated Autonomous AI Database Deployments to learn about using Oracle Database features and options in Autonomous AI Databases.

Download, Install and Configure Oracle Database JDBC Driver

Follow these steps to download and install Oracle Database JDBC Driver and then download the client credentials for your Autonomous AI Database and make 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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as follows. If you don't have such an account, you will need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

- a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous AI 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.

Download, Install and Configure Oracle Instant Client

Follow these steps to download and install Oracle Instant Client and then download the client credentials for your Autonomous AI Database and make 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
```
 - **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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as follows. If you don't have such an account, you need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

- a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous AI 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.

The Basics of Building an Application

Autonomous AI Database on Dedicated Exadata Infrastructure supports application development in a wide variety of programming languages and platforms.

In general, regardless of the language you use to build an application, you follow the same guidelines to build an application that takes advantage of the high-performance and continuous-availability features of an Autonomous AI Database:

- **Connect through an Oracle client.** When you connect to the database through an Oracle client, almost all connection-management operations are performed by the client, permitting you to concentrate on the business logic of your application. Depending on your programming language, you use the Oracle Database JDBC Driver or the Oracle Instant Client.
- **Use connection pools.** When you code your application to use connection pools instead of creating and destroying connections individually, you gain performance improvements and permit the Oracle client you connect through to perform continuous-availability

operations automatically. How you code to use connection pools depends on your programming language.

- **Connect to the appropriate database service.** Autonomous AI Database provides several pairs of database services to use when connecting to your database. These pairs of database connection services are designed to support different kinds of database operations, as described in *Predefined Database Service Names for Autonomous AI Databases*.

Also regardless of the language you use to build an application, you perform the same basic tasks to configure your system to support application development:

1. Download and install the basic software to develop in the given language. For example, you download and install JDK to develop Java applications.
2. Download and install any extension library or module necessary to permit applications in the given language to connect to an Oracle Database and make SQL calls. For example, you download and install the `cx_Oracle` extension module to develop Python applications.
3. Download and install the Oracle client appropriate for the given language and extension library or module.
4. Download the client credentials for the database and make them available to Oracle client you installed.

For instructions on performing the last two steps for Oracle Database JDBC Driver, see *Prepare for JDBC Thin Connections*. For instructions on performing these steps for Oracle Instant Client, see *Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections*.

For some languages and platforms, you can follow specific step-by-step instructions instead of the general guidelines, as discussed below.

Build Database Applications with Autonomous AI Database

Build a Java Application

To build a Java application that accesses an Autonomous AI Database, you start by configuring your development system to support database access that can take advantage of the continuous availability and high performance features of Autonomous AI Database.

After configuring your development system to support database access, you code database connections and SQL statements in your application to take advantage of the continuous availability and high performance features.

✓ Tip

For a "try it out" alternative to reading the following topics, you can go through the *Lab 5: Build Java Application Stacks* in *Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop*.

Configure Your Java Development System

To configure your development system so that your Java application can take advantage of the continuous availability and high performance features of an Autonomous AI Database, perform these steps.

1. Download and install the Java Development Kit (JDK).

2. Download the client credentials for your Autonomous AI Database.
3. Get the Oracle Java Database Connectivity (JDBC) drivers.

Before You Begin

Your development system must meet certain criteria to configure it successfully.

- It must have internet access.
- It must have network access to the Autonomous AI Database.

See [Configure a Development System to Access the Database](#) to learn how to ensure your development system meets this requirement.

Download and Install the JDK

Go to the [Java SE Downloads](#) page. Then, download and install JDK 8u221 or later by following the instructions on the page.

Download the Client Credentials for Your Autonomous AI Database

1. 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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as follows. If you don't have such an account, you need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

- a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous AI Database.
- b. Click **DB Connection**.
- c. On the **Database Connection** page click **Download**.
- d. In the **Download Wallet** dialog, enter a 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** and unzip, to save the client credentials zip file to a secure directory.

Get the Oracle JDBC Drivers

Get the Oracle JDBC drivers, version 19.6.0.0 or later, from *either* Maven Central *or* the [JDBC Downloads](#) page at Oracle Technical Resources. (See the Oracle Technologies [JDBC Home page](#) for related videos and other resources.)

To get the JDBC drivers from Maven Central, follow these steps.

1. Get the Oracle JDBC drivers from [Central Maven Repository](#).

Provide the driver Maven dependency GAV (GroupID, ArtifactID, VersionID), to pull `ojdbc8.jar`, along with other jars such as `oraclepki.jar`, `osdt_core.jar`, and `osdt_cert.jar`. See [Maven Central Guide](#).

For ojdbc8.jar version 19.6.0.0, provide this GAV:

```
<groupId>com.oracle.database.jdbc</groupId>
<artifactId>ojdbc8</artifactId>
<version>19.7.0.0</version>
```

For ojdbc8.jar version 19.7.0.0, provide this GAV:

```
<groupId>com.oracle.database.jdbc</groupId>
<artifactId>ojdbc8-production</artifactId>
<version>19.7.0.0</version>
<type>POM</type>
```

To get the JDBC drivers from Oracle Technical Resources, follow these steps.

1. Go to the Oracle JDBC [Downloads](#) page. Then, choose the latest version of the drivers to go to its version-specific download page.
2. Download and unzip this archive to the directory where you want to place the JDBC driver: ojdbc8-full.tar.gz.
3. Point the connection URL to your Autonomous AI Database.

Append TNS_ADMIN to the connection URL, setting its value to the full path of the directory where you unzipped the client credentials. For example:

```
// Use TNS alias name plus TNS_ADMIN with JDBC driver 18.3 or higher
DB_URL="jdbc:oracle:thin:@wallet_dbname?
TNS_ADMIN=/Users/test/wallet_dbname";
```

```
// For Microsoft Windows, use this for TNS_ADMIN:
// TNS_ADMIN=C:\\Users\\test\\wallet_dbname";
```

4. Add the paths to the following unzipped JAR files to the CLASSPATH environment variable you use when you compile and run Java programs.

Use [DataSourceSample.java](#) or [UCPSample.java](#) to verify the connection to your Autonomous AI Database.

- ojdbc8.jar: the core JDBC driver
- oraclepki.jar, osdt_core.jar, and osdt_cert.jar: for an Autonomous AI Database that uses wallet-based authentication
- ucp.jar: for Universal Connection Pooling (UCP)
- ons.jar and simplefan.jar: for FAN (Fast Application Notification) support

Code Database Connections and SQL Statements

After configuring your development system to support Java application connectivity to an Autonomous AI Database, follow these guidelines to achieve high performance and continuous availability of your application's connections to the database:

- Use connection pools.
- Enable FAN (Fast Application Notification).

- Use the predefined database service that best matches the operations you will be performing. For information about the predefined database services, see *Predefined Database Service Names for Autonomous AI Databases*.

For example:

```
import java.sql.Connection;
import javax.sql.PooledConnection;
import oracle.jdbc.OracleConnection;
import oracle.jdbc.replay.OracleDataSourceFactory;
import oracle.jdbc.replay.OracleDataSource;
import oracle.jdbc.replay.OracleConnectionPoolDataSource;
...
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
// Set the connection factory first before all other properties
pds.setConnectionFactoryClassName(
    "oracle.jdbc.replay.OracleConnectionPoolDataSourceImpl");
pds.setFastConnectionFailoverEnabled(true);
pds.setURL("jdbc:oracle:thin:@tp_tls?TNS_ADMIN=/users/jdoe/adbcredentials");
pds.setUser("appuser");
pds.setPassword("<password>");
pds.setConnectionPoolName("JDBC_UCP_POOL");

Connection conn = pds.getConnection();
```

Additional Resources

For detailed information about the Oracle Database JDBC Driver, see:

- *Introducing JDBC* in Oracle Database 19c JDBC Developer's Guide or [Oracle Database 26ai JDBC Developer's Guide](#)
- *Oracle Database JDBC Java API Reference* in Oracle Database 19c JDBC Developer's Guide or [Oracle Database 26ai JDBC Developer's Guide](#)

For detailed information about the Universal Connection Pool, see:

- *Introduction to UCP* in Oracle Database 19c Universal Connection Pool Developer's Guide or [Oracle Database 26ai Universal Connection Pool Developer's Guide](#)
- *Oracle Universal Connection Pool API Reference* in Oracle Database 19c Universal Connection Pool Java API Reference or [Oracle Database 26ai Universal Connection Pool Java API Reference](#)

Build a .NET Application with Visual Studio and Visual Studio Code

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure supports connectivity to the Microsoft .NET Framework, .NET Core, Visual Studio, and Visual Studio Code.

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

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

Oracle Developer Tools for Visual Studio provides database application design-time support in Visual Studio, including tools for managing Autonomous AI Databases in Oracle Cloud.

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

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

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

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

Set-Up Instructions

Refer to [Developing .NET Applications for Oracle Autonomous AI Database](#) for instructions on how to download, install, and configure these components for use with Oracle Autonomous AI Database on Dedicated Exadata Infrastructure.

Note that these instructions show how to make TCPS (TCP with SSL) connections to your database, which requires using wallets. For databases on dedicated infrastructure, you can instead make TCP connections to the database, which does not require use of a wallet.

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

Configure for High Availability and Performance

In the newest ODP.NET versions, Fast Connection Failover (FCF) and Connection Runtime Load Balancing (RLB) are enabled automatically via the connection string default settings. FCF requires the connection string settings, Pooling (true) and HA Events (true). RLB requires the settings, Pooling (true) and Load Balancing (true).

Application Continuity is currently only available in unmanaged ODP.NET. It is enabled by default via the connection pool setting, Application Continuity (true).

No additional configuration is necessary when using TCP for ONS connections.

ONS TCPS requires some configuration and code changes depending on the provider used:

- **ODP.NET Core.** To configure ONS TCPS for ODP.NET Core, use `OracleConfiguration.OnsWalletLocation` to enable TPCS and indicate the wallet location.

```
OracleConfiguration.OnsWalletLocation = @"<wallet-directory>";
```

- **ODP.NET Managed Driver.** To configure ONS TCPS for managed ODP.NET, modify the `oracle.manageddataaccess.client` configuration section similar to the sample below.

```
<oracle.manageddataaccess.client>
  <version number="*">
    <onsConfig mode="remote">
      <settings>
        <setting name="Protocol" value="TCPS" />
        <setting name="WALLET_LOCATION" value="<wallet-directory>" />
      </settings>
      <ons database="<database-name>">
        <add name="nodeList" value="slcai611:6205,slcai610:6205,slcai612:6205" />
      </ons>
    </onsConfig>
  </settings>
```

```

        <setting name="WALLET_LOCATION" value="<wallet-directory>" />
      </settings>
    </version>
  </oracle.manageddataaccess.client>

```

- **ODP.NET Unmanaged Driver.** To configure ONS TCPS for unmanaged ODP.NET, modify the oracle.unmanageddataaccess.client configuration section similar to the sample below.

```

<oracle.unmanageddataaccess.client>
  <version number="*">
    <onsConfig mode="remote">
      <settings>
        <setting name="Protocol" value="TCPS" />
        <setting name="WALLET_LOCATION" value="<wallet-directory>" />
      </settings>
      <ons database="<database-name>">
        <add name="nodeList" value="slcai611:6205,slcai610:6205,slcai612:6205" />
      </ons>
    </onsConfig>
  </version>
</oracle.unmanageddataaccess.client>

```

Build a Node.js Application

To build a Node.js application that accesses an Autonomous AI Database, you start by configuring your development system to support database access that can take advantage of the continuous availability and high performance features of Autonomous AI Database.

After configuring your development system, you code database connections and SQL statements in your application to take advantage of the continuous availability and high performance features.

✓ Tip

For a "try it out" alternative to reading the following topics, you can go through the *Lab 3: Build and Deploy Node.js Application* in Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop.

Configure Your Node.js Development System

To configure your development system so that your Node.js application can take advantage of the continuous availability and high performance features of an Autonomous AI Database, you perform these steps.

1. Download and install Node.js.
2. Download and install Oracle Instant Client.
3. Download and install node-oracledb.
4. Download the client credentials for the database and make them available to Oracle Instant Client.

Before You Begin

Your development system must meet certain criteria to configure it successfully.

- It must have internet access.
- It must have network access to the Autonomous AI Database.

See [Configure a Development System to Access the Database](#) to learn how to ensure your development system meets this requirement.

Download and Install Node.js

Download and install Node.js for your system's OS and architecture:

- **Oracle Linux:**

Run these commands to download and install the latest version of Node.js:

```
sudo yum install -y oracle-release-el7 oracle-nodejs-release-el7
sudo yum install -y nodejs
```

- **Other OSES and architectures:**

Go to the Node.js [Downloads](#) page, select the latest LTS (Long Term Support) version for your system's OS and architecture, and install it.

Download and Install Oracle Instant Client

You need Oracle Instant Client libraries version 19.6 or later.

Download and install the Oracle Instant Client basic package for your system's OS and architecture:

- **Oracle Linux:**

Run these commands to download and install the Oracle Instant Client basic package:

```
sudo yum -y install oracle-release-el7
sudo yum -y install oracle-instantclient19.3-basic
```

(If you want to see a list of all Instant Client packages, go to http://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient/x86_64/index.html.)

- **Other OSES and architectures:**

1. Go to the [Oracle Instant Client Downloads](#) page and select the download for your system's OS and architecture.
2. 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.

Download and Install node-oracledb

Download and install the node-oracledb add-on for Node.js for your system's OS and architecture:

- **Oracle Linux:**

Run these commands to download and install the latest version of node-oracledb:

```
sudo yum install -y oracle-release-el7 oracle-nodejs-release-el7
sudo yum install -y node-oracledb-node10
```

- **Other OSES and architectures:**

Go to the [Installing node-oracledb](#) page, choose the "My database is on another machine" instructions for your OS and architecture, and then follow the **Install the add-on** instructions.

Download and Install Client Credentials for the Database

1. 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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as follows. If you don't have such an account, you need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

- a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous AI 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.
2. 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/adbccredentials")
```

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

Code Database Connections and SQL Statements

After configuring your development system to support Node.js application connectivity to an Autonomous AI Database, follow these steps to ensure optimal performance of your application's use of the database:

1. Add the dependency on the `node-oracledb` add-on to your application's `package.json` file.
2. Code connections for high performance and continuous availability.

Add the `node-oracledb` Dependency to `package.json`

Edit the `dependencies` object in the `package.json` file for your application, adding the `oracledb` package and version. (Use command `npm init` to generate `package.json` if it doesn't exist.) For example:

```
...
"dependencies": {
```

```

    . . .
    "oracledb": "^4.0",
    . . .
  },
  . . .

```

For detailed information about the dependencies object, see the [npm-package.json](#) page. To display the oracledb version installed, you can use the `npm list` command; for example:

```
npm list -g --depth=0
```

Code Connections for High Performance and Continuous Availability

To achieve high performance and continuous availability, follow these guidelines when making connections to the database:

- Use connection pools.
- Enable FAN (Fast Application Notification).
- Use the predefined database service that best matches the operations you will be performing. For information about the predefined database services, see [Predefined Database Service Names for Autonomous AI Databases](#).

For example:

```

pool = await oracledb.createPool({
  events: true,
  user: "appuser",
  password: process.env.MY_PASSWORD_ENV_VAR,
  connectString: "tp_tls"
});

```

This example creates a pool for connections to the `tp_tls` database service, and enables FAN by setting the `events` property to `true`.

Additional Resources

For detailed information about node-oracledb, go to the [node-oracledb Documentation](#) page, which includes both an API Reference and a User Guide.

For code examples that demonstrate a wide variety of node-oracledb features, go to the [node-oracledb examples](#) folder.

Build a PHP Application

To build a PHP application that accesses an Autonomous AI Database, you start by configuring your development system to support database access that can take advantage of the continuous availability and high performance features of Autonomous AI Database.

After configuring your development system, you code database connections and SQL statements in your application to take advantage of the continuous availability and high performance features.

Configure Your PHP Development System

To configure your development system so that your PHP application can take advantage of the continuous availability and high performance features of an Autonomous AI Database, you perform these steps.

1. Download and install PHP.
2. Download and install Oracle Instant Client.
3. Download and install PHP OCI8.
4. Download the client credentials for the database and make them available to Oracle Instant Client.

Before You Begin

Your development system must meet certain criteria to configure it successfully.

- It must have internet access.
- It must have network access to the Autonomous AI Database.

See [Configuring a Development System to Access the Database](#) to learn how to ensure your development system meets this requirement.

Download and Install PHP

Download and install PHP for your system's OS and architecture:

- **Oracle Linux:**

Run these commands to download and install PHP and to download and install PEAR to get PHP's pecl package management command:

```
sudo dnf install -y oracle-release-el8 oracle-php-release-el8
sudo dnf install -y php php-devel php-xml dtrace-utils
wget http://pear.php.net/go-pear.phar
sudo php go-pear.phar
```

- **Other OSES and architectures:**

Go to the PHP [Installation and Configuration](#) page, and follow the instructions for your system's OS and architecture.

Download and Install Oracle Instant Client

Download and install the Oracle Instant Client basic package for your system's OS and architecture:

- **Oracle Linux:**

Run these commands to download and install the Oracle Instant Client basic package:

```
sudo dnf -y install oracle-release-el8
sudo dnf -y install oracle-instantclient21.13-basic oracle-instantclient21.13-devel
```

(If you want to see a list of all Instant Client packages, go to http://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient/x86_64/index.html.)

- **Other OSES and architectures:**

1. Go to the [Oracle Instant Client Downloads](#) page and select the download for your system's OS and architecture.
2. On the download page, accept the Oracle Technology Network License Agreement, download the latest versions of the **Basic Package** and the **SDK Package**, and then install them by following the instructions at the bottom of the download page.

Download and Install PHP OCI8

Download and install the OCI8 extension for PHP for your system's OS and architecture:

- **Oracle Linux:**

Run these commands to download and install the latest version of the OCI8 extension:

```
sudo PHP_DTRACE=yes pecl install oci8
sudo sh -c "echo extension=oci8.so > /etc/php.d/20-oci8.ini"
sudo sh -c "echo oci8.events = 0n > /etc/php.d/20-oci8.ini"
```

If your applications needs Oracle environment variables, add them to the `/etc/sysconfig/httpd` config file, for example:

```
NLS_LANG=AMERICAN_AMERICA.AL32UTF8
NLS_DATE_FORMAT='YYYY-MM-DD HH24:MI:SS'
ORA_SDTZ=UTC
```

If you add such variables, restart the Apache HTTP server to read the new variables:

```
sudo systemctl restart httpd
```

- **Other OSes and architectures:**

1. Run this command

```
pecl install oci8
```

2. Add the following lines to your PHP installation's `php.ini` file:

```
extension=oci8.so
oci8.events = 0n
```

Download and Install Client Credentials for the Database

1. 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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as follows. If you don't have such an account, you need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

- a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous AI 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.
2. 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.

Code Database Connections and SQL Statements

After configuring your development system to support PHP application connectivity to an Autonomous AI Database, follow these guidelines to achieve high performance and continuous availability of your application's connections to the database:

- Use connection pools.
- Enable FAN (Fast Application Notification).
- Use the predefined database service that best matches the operations you will be performing. For information about the predefined database services, see [Predefined Database Service Names for Autonomous AI Database](#).

For PHP, you enable FAN when you add the line `oci8.events = On` to the `php.ini` file as part of downloading and installing the Oracle OCI8 extension to PHP.

The OCI8 interface does not expose specific connection pooling calls. Instead, it uses connection pooling internally. To realize the benefits of connection pooling, use the `oci_pconnect()` function to create or reuse a persistent connection that remains across HTTP requests.

For example:

```
$connection = oci_pconnect("appuser", getenv("MY_PASSWORD_ENV_VAR"),  
"tp_tls");
```

This example creates a persistent connection to the `tp_tls` database service.

Additional Resources

For information about the Oracle OCI8 extension to PHP, see the [Oracle OCI8](#) page.

For information about using PHP with Oracle Database, see [The Underground PHP and Oracle Manual](#).

Build a Python Application

To build a Python application that accesses an Autonomous AI Database, you start by configuring your development system to support database access that can take advantage of the continuous availability and high performance features of Autonomous AI Database.

After configuring your development system, you code database connections and SQL statements in your application to take advantage of these features.

✓ Tip

For a "try it out" alternative to reading the following topics, you can go through the *Lab 4: Build Python Application Stacks* in Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop.

Configure Your Python Development System

To configure your development system so that your Python application can take advantage of the continuous availability and high performance features of an Autonomous AI Database, you perform these steps.

1. Download and install Python.
2. Download and install Oracle Instant Client.
3. Download and install cx_Oracle.
4. Download the client credentials for the database and make them available to Oracle Instant Client.

Before You Begin

Your development system must meet certain criteria to configure it successfully.

- It must have internet access.
- It must have network access to the Autonomous AI Database.

See [Configure a Development System to Access the Database](#) to learn how to ensure your development system meets this requirement.

Download and Install Python

- **Oracle Linux:**

Oracle Linux 7 includes Python 2.7, so you simply run this command::

```
sudo yum -y install oracle-release-el7
```

- **Other OSes and architectures:**

Go to the [python.org Downloads](https://python.org/downloads) page and download and install the latest Python 2.7 or Python 3.5 (or later) version for your OS and architecture.

Download and Install Oracle Instant Client

You need Oracle Instant Client libraries version 19.6 or later.

Download and install the Oracle Instant Client basic package for your system's OS and architecture:

- **Oracle Linux:**

Run these commands to download and install the Oracle Instant Client basic package:

```
sudo yum -y install oracle-release-el7
sudo yum -y install oracle-instantclient19.3-basic
```

(If you want to see a list of all Instant Client packages, go to http://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient/x86_64/index.html.)

- **Other OSes and architectures:**

1. Go to the [Oracle Instant Client Downloads](#) page and select the download for your system's OS and architecture.
2. 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.

Download and Install cx_Oracle

Use Python's pip package to install cx_Oracle from PyPI (the Python Package Index):

- **Oracle Linux:**

Run these commands to download the pip package and then use it to install cx_Oracle:

```
sudo yum -y install oracle-release-el7
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
sudo python get-pip.py
python -m pip install cx_Oracle --upgrade
```

- **Other OSes and architectures:**

Run this command:

```
python -m pip install cx_Oracle --upgrade
```

Download and Install Client Credentials for the Database

1. 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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as follows. If you don't have such an account, you need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

- a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous AI 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.
2. 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/adbccredentials")
```

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

Code Database Connections and SQL Statements

After configuring your development system to support Python application connectivity to an Autonomous AI Database, follow these guidelines to achieve high performance and continuous availability of your application's connections to the database:

- Use connection pools.
- Enable FAN (Fast Application Notification).
- Use the predefined database service that best matches the operations you will be performing. For information about the predefined database services, see [Predefined Database Service Names for Autonomous AI Database](#).

For example:

```
pool = cx_Oracle.SessionPool("appuser",
                             SampleEnv.GetMainPassword(),
                             "tp_tls",
                             events=True,
                             threaded=True)
```

This example creates a pool for connections to the tp_tls database service, and enables FAN by setting the events parameter to True.

Additional Resources

For detailed information about cx_Oracle, go to the [cx_Oracle Documentation](#) page.

For code examples that demonstrate a wide variety of cx_Oracle features, go to the [python-cx_Oracle samples](#) folder.

Build a Ruby Application

To build a Ruby application that accesses an Autonomous AI Database, you start by configuring your development system to support database access that can take advantage of the continuous availability and high performance features of Autonomous AI Database.

After configuring your development system to support database access, you code database connections and SQL statements in your application to take advantage of these features.

Configure Your Ruby Development System

To configure your development system so that your Ruby application can take advantage of the continuous availability and high performance features of an Autonomous AI Database, you perform these steps.

1. Download and install Ruby.
2. Download and install Oracle Instant Client.
3. Download and install ruby-oci8.
4. Download the client credentials for the database and make them available to Oracle Instant Client.

Before You Begin

Your development system must meet certain criteria to configure it successfully.

- It must have internet access.
- It must have network access to the Autonomous AI Database.

See [Configure a Development System to Access the Database](#)

Download and Install Ruby

To download and install Ruby, go to the [Installing Ruby](#) page at ruby-lang.org and follow the instructions for your system's OS and architecture.

Download and Install Oracle Instant Client

Download and install the Oracle Instant Client basic package for your system's OS and architecture:

- **Oracle Linux:**

Run these commands to download and install the Oracle Instant Client basic package:

```
sudo yum -y install oracle-release-el7
sudo yum -y install oracle-instantclient19.3-basic
```

(If you want to see a list of all Instant Client packages, go to http://yum.oracle.com/repo/OracleLinux/OL7/oracle/instantclient/x86_64/index.html.)

- **Other OSes and architectures:**

1. Go to the [Oracle Instant Client Downloads](#) page and select the download for your system's OS and architecture.
2. 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.

Download and Install ruby-oci8

To download and install the ruby-oci8 interface to Oracle Database, follow the instructions in the [Install for Oracle Instant Client](#) page.

Download and Install Client Credentials for the Database

1. 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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as follows. If you don't have such an account, you need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

- a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous AI 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.

2. 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/adbccredentials")
```

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

Code Database Connections and SQL Statements

After configuring your development system to support Ruby application connectivity to an Autonomous AI Database, follow these guidelines to achieve high performance and continuous availability of your application's connections to the database:

- Use connection pools.
- Enable FAN (Fast Application Notification).
- Use the predefined database service that best matches the operations you will be performing. For information about the predefined database services, see [Predefined Database Service Names for Autonomous AI Databases](#).

For example:

```
require 'oci8'
OCI8.properties[:events_mode] = true
cpool = OCI8::ConnectionPool.new(1, 5, 2, username, password, 'tp_tls')
conn1 = OCI8.new(username, password, cpool)
```

This example enables FAN by setting the `events_mode` property to `true`, creates a pool for connections to the `tp_tls` database service, and acquires a connection from the pool.

Additional Resources

For detailed information about `ruby-oci8`, go to the [Ruby-oci8](#) page.

Build an APEX (Oracle Application Express) Application

Autonomous AI Database on Dedicated Exadata Infrastructure supports application development with Oracle Application Express (APEX). Oracle APEX on Autonomous AI Database provides a preconfigured, fully managed, and secured environment to build and deploy world-class data-centric applications.

Oracle APEX provides you with an easy-to-use browser-based environment to load data, manage database objects, develop REST interfaces, and build applications. There are no limits on the number of developers or end-users for your Oracle APEX applications. Autonomous AI Database can instantly scale compute and storage online as needed, based upon your workload. Additionally, you can deploy the Oracle APEX applications developed on-premise to Oracle APEX on Autonomous AI Database, or vice-versa easily.

Configuration, patching, monitoring, and upgrading all Oracle Application Express components are fully managed by Oracle. Thereby, you are 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 with less cost and greater consistency. You can adapt to changing requirements with ease. You can empower professional developers and everyone else in your organization to be a part of the solution.

✓ **Tip**

For a "try it out" alternative that demonstrates these instructions, run *Lab 8: Build APEX Application* in the [Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop](#).

Access Oracle Application Express in Autonomous AI Database

📘 **Note**

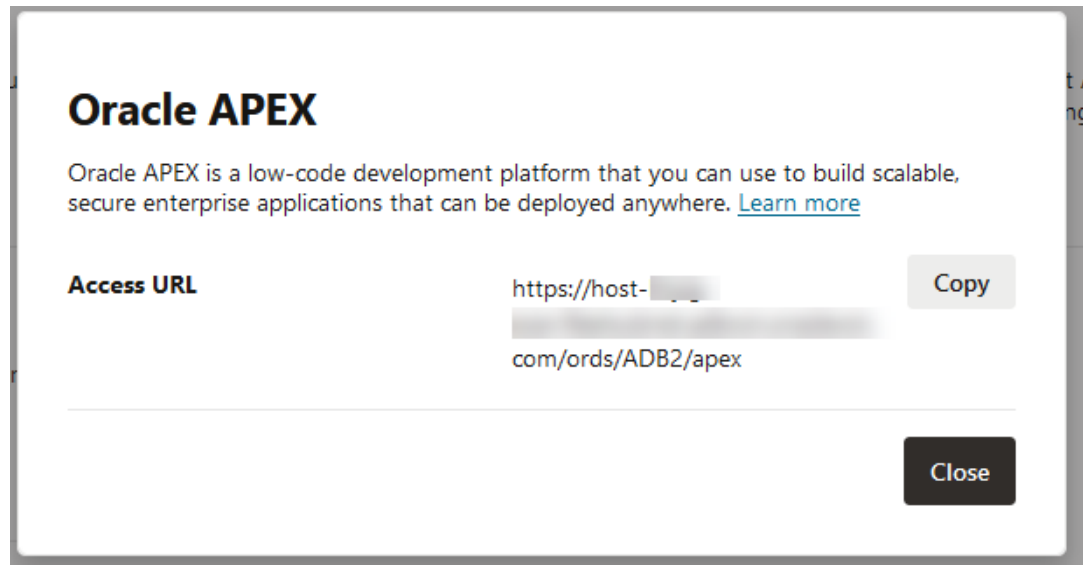
To access Oracle Applications Express, the system you are using must have network access to the Autonomous AI Database, as described in [Configure a Development System to Access the Database](#).

Learn how to access Oracle Application Express from Autonomous AI Database on Dedicated Exadata Infrastructure.

By default, only the ADMIN database user has access to Oracle Application Express. Before you can sign in as another database user, the ADMIN user must create a workspace and enable access to the workspace to that user.

To access Oracle Application Express:

1. In your web browser, sign in to Oracle Cloud and navigate to the **Details** page for the Autonomous AI Database. See [View Details of a Dedicated Autonomous AI Database](#).
2. Open the **Tools** tab.
3. Click **Open APEX** in the **Oracle Application Express** tile.
4. In the **Oracle Application Express** dialog, click the **Copy** link in the **Access URL** field to copy the URL to your clipboard.



5. Paste the access URL into your web browser's address bar to access Oracle Application Express.

Note

As Oracle Application Express uses a private IP address that is only accessible from a Compute instance running with the virtual cloud network (VCN) used by your Autonomous AI Database, you must paste the access URL into a browser that is running on a compute instance in the VCN.

6. Enter the following details, and click **Sign In**:
 - **Workspace**, to which the ADMIN user enabled access to this database user.
 - **Username**, the database user with which you are trying to access Oracle APEX.
 - **Password** of the database user.
7. Upon signing in successfully, you will see the Oracle Application Express home page with the following features:
 - **App Builder** enables you to design, develop, and deploy beautiful and responsive web applications.
 - **SQL Workshop** lets you access your Autonomous AI Database objects, run SQL queries on them, execute SQL scripts, and load data.
 - **Team Development** helps you keep track of tasks, features, bugs and feedback for your entire team.
 - **App Gallery** provides a list of sample and productivity apps that you can install and leverage for your application development.

Access Oracle Application Express Administration Services

Note

To access Oracle Applications Express, the system you are using must have network access to the Autonomous AI Database, as described in [Configure a Development System to Access the Database](#).

Learn how to access Oracle Application Express administration services from an Autonomous AI Database and create a workspace.

To access Oracle Application Express administration services as the ADMIN database user:

1. Copy the Oracle APEX access URL from your an Autonomous AI Database **Details** page, and paste it in your web browser's address bar. For more details, see [Access Oracle Application Express in Autonomous AI Database](#).

Note

As Oracle Application Express uses a private IP address that is only accessible from a Compute instance running with the virtual cloud network (VCN) used by your Autonomous AI Database, you must paste the access URL into a browser that is running on a compute instance in the VCN.

2. On the **Sign In** page, scroll down to view **Tasks**.
3. Click **Administration** to sign in to administration services.
4. Enter the following details, and click **Sign In**:
 - **Username:** `ADMIN`
 - **Password:** The ADMIN password that you defined while creating the Autonomous AI Database.
5. Upon signing in successfully, you will see the Oracle Application Express Administration Services home page with different administration options to create and manage workspaces and users, and enable user access to different workspaces.

Note

If you're signing into Oracle Application Express administration services for the first time, you will be prompted to create a workspace. Click the **Create Workspace** button in the dialog box and provide required details to create the first workspace.

6. To create more workspaces, click the **Create Workspace** button from the home page and provide the workspace details as prompted by the wizard.
7. To create Oracle APEX developer accounts:
 - Click the **Manage Workspaces** tile from the home page.
 - Click **Manage Developers and Users** under **Workspace Actions**.
 - Click **Create User** and enter the new developer user details.

See *Creating User Accounts in Oracle Application Express Administration Guide* for more information.

Upgrade Oracle Application Express Instance

Autonomous AI Database on Dedicated Exadata Infrastructure supports Oracle Application Express 24.2 on all the newly provisioned Autonomous Container Databases (ACDs). However, all existing ACDs continue to run with the Application Express version they were provisioned with.

If you want to use the latest Oracle Application Express features for enhancing the existing applications, schedule an upgrade to your Oracle Application Express instance through Service Request (SR) in My Oracle Support. You must supply the OCIDs of the tenancy and Autonomous AI Database for which you request the APEX upgrade in the SR.

See *Create a Service Request in My Oracle Support* to learn how to contact Oracle Support for assistance.

Limitations on the Use of Oracle Application Express

This section lists the feature restrictions and limitations of Oracle Application Express when used within the context of an Autonomous AI Database.

The following limitations are required to protect the security and performance of your Oracle Application Express environment:

- LDAP Directory application authentication scheme is not supported.
- Oracle Application Express is only available as a Full Development environment. Converting into a Runtime environment, which minimizes the installed software footprint and removes UI components such as App Builder and Administration Services, is not supported.

Developing RESTful Services in Autonomous AI Database

You can develop and deploy RESTful Services with native Oracle REST Data Services (ORDS) support on an Autonomous AI Database.

About Oracle REST Data Services in Autonomous AI Database

Oracle REST Data Services (ORDS) makes it easy to develop REST interfaces for relational data in an Autonomous AI Database. ORDS is a mid-tier Java application that maps HTTP(S) verbs, such as GET, POST, PUT, DELETE, and so on, to database transactions, and returns any results as JSON data.

Note

The Oracle REST Data Services (ORDS) application in Autonomous AI Database is preconfigured and fully managed. ORDS connects to the database using the low predefined database service with a fixed maximum number of connections (the number of connections for ORDS does not change based on the number of CPUs). It is not possible to change the default ORDS configuration.

See [Oracle REST Data Services](#) for information on using Oracle REST Data Services.

See [Predefined Database Service Names for Autonomous AI Databases](#) for information on the low database service.

Develop with Oracle REST Data Services on Autonomous AI Database

Autonomous AI Database 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 AI Database and enable REST services access to tables and views, or develop custom RESTful Services based on your SQL and PL/SQL code. See [Connect with Oracle SQL Developer 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 AI Database ADMIN account is REST Enabled. This allows for REST Services to be published in the ADMIN schemas and allows you to access Database Actions using the ADMIN database user account. Oracle recommends you create an application schema account for your RESTful Services and REST enabled objects. Services are secured using Database Authentication and your REST enabled schema.

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

```
GET /ords/hr/module/service/
```

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

```
GET /ords/hr/module/service/
```

results in an error:

```
401 HTTP Unauthorized response/error
```

Any database user whose credentials are correct and meets these rules is authenticated and granted the ORDS, mid-tier, role: SQL Developer. The SQL Developer role enables the user to access any endpoint that requires the SQL Developer role.

See [REST-Enable a Database Table](#) in *Quick Start Guide* for information on how to enable a table for REST access.

Send Email and Notifications from Autonomous AI Database

Send Email on Autonomous AI Database on Dedicated Exadata Infrastructure

You can send email on Autonomous AI Database on Dedicated Exadata Infrastructure using the Email Delivery Service or the DBMS_CLOUD_NOTIFICATION procedures. This article describes how to send email using both these options.

Send Email with Email Delivery Service on Autonomous AI Database

Describes the steps to send email using UTL_SMTP on Autonomous AI Database on Dedicated Exadata Infrastructure.

To send email using UTL_SMTP:

1. Identify your SMTP connection endpoint for Email Delivery.
 - a. If you want to use Oracle Cloud Infrastructure (OCI) Email Delivery Service, identify your connection endpoint for Email Delivery. For example, select smtp.email.us-phoenix-1.oci.oraclecloud.com when US West (Phoenix) is the sending region. See from [Configure SMTP Connection](#) for more information.

Note

You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is unavailable in your current region.

- b. If you want to use a third-party SMTP server, your network administrator must configure the SMTP server and its connectivity with the Autonomous AI Database tenant and provide the SMTP connection endpoint information.
2. Generate SMTP credentials for Email Delivery for the OCI user.

UTL_SMTP uses credentials to authenticate with Email Delivery servers when you send an email. See [Generate SMTP Credentials for a User](#) for more information.
3. Create approved senders for Email Delivery.

If you are using OCI Email Delivery Service, complete this step for all email addresses you use as the "From" with UTL_SMTP.MAIL. See [Managing Approved Senders](#) for more information.
4. Allow SMTP host access for the user(s) with administrator role by appending Access Control Entry (ACE).

To use secure SMTP, you must set up the database user with Access Control Entry (ACE) for wallet access, in addition to SMTP host access. For secure SMTP access using OCI Email Delivery Service, the Oracle Cloud Infrastructure account must be authenticated using Simple Authentication and Security Layer (SASL) protocols such as PLAIN, CRAM-MD5, LOGIN. For example, see the sample code in [SMTP Send Email Sample Code](#).

5. Write the PL/SQL code to send email using UTL_MAIL.

Tip

If you are developing the send email procedure by yourself, ensure that your code is according to [SMTP Send Email Sample Code](#).

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

For example:

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

See *UTL_SMTP* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for further reference.

See Notes on the Use of Database PL/SQL Packages for more information about using UTL_SMTP with Autonomous AI Database.

SMTP Send Email Sample Code

Shows sample code for sending email with UTL_SMTP on an Autonomous AI Database.

Sample Code

```
-- As ADMIN user, set up ACE to access SMTP host
connect admin/[password]@[ADBD-host]/[pdbname]_tp.atp.oraclecloud.com
begin
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
    host      => '[SMTP_Server_address]',
    lower_port => 25,
    upper_port => 25,
    ace       => xs$ace_type(
      privilege_list => xs$name_list('smtp'),
      principal_name => '[database_user]',
      principal_type => xs_acl.ptype_db));
end;

/

-- If using secure SMTP, set up ACE to wallet for certificate-based secure (TLS) email
declare
  m_wallet_path VARCHAR2(512);
begin
  select property_value into m_wallet_path
    from database_properties
   where property_name like '%SSL%';
  m_wallet_path := 'file:' || m_wallet_path;
  DBMS_NETWORK_ACL_ADMIN.APPEND_WALLET_ACE (
    wallet_path => m_wallet_path,
    ace         => xs$ace_type(
      privilege_list => xs$name_list('use_client_certificates'),
      principal_name => '[database_user]',
      principal_type => xs_acl.ptype_db));
end;

/

-- As ADMIN or regular database user with privileges set up above, use UTL_SMTP to send email
connect [database_user]/[password]@[ADBD-host]/[pdbname]_tp.atp.oraclecloud.com

-- Example of normal (non-secure) email
DECLARE
  c          UTL_SMTP.CONNECTION;
  reply      UTL_SMTP.REPLY;
  host       varchar2(64) := '[SMTP_Server_address]',
  port       pls_integer := 25;
  sender     varchar2(64) := '[sender-address@domain.com]';
  recipient  varchar2(64) := '[recipient-address@domain.com]';
BEGIN
  c := UTL_SMTP.OPEN_CONNECTION (host, port);
  if (reply.CODE != 220) then
    raise_application_error(-20000, 'UTL_SMTP.CONNECTION: '
      || reply.CODE || ' - ' || reply.TEXT);
  end if;
```

```

UTL_SMTP.HELO(c, host);
UTL_SMTP.MAIL(c, sender);
UTL_SMTP.RCPT(c, recipient);
UTL_SMTP.OPEN_DATA(c);
UTL_SMTP.WRITE_DATA(c, 'From:[Sender] <' || sender || '>' || UTL_TCP.CRLF);
UTL_SMTP.WRITE_DATA(c, 'To:[Recipient] <' || recipient || '>' || UTL_TCP.CRLF);
UTL_SMTP.WRITE_DATA(c, 'Subject: Email from UTL_SMTP App' || UTL_TCP.CRLF);
UTL_SMTP.WRITE_DATA(c, 'Mail Content');
UTL_SMTP.CLOSE_DATA(c);
dbms_output.put_line('Close Data: ' || reply.code);
UTL_SMTP.QUIT(c);
END;
/

-- Example of secure email using OCI Mail Server as the SMTP server
DECLARE
  m_wallet_path VARCHAR2(256);
  c              UTL_SMTP.CONNECTION;
  reply          UTL_SMTP.REPLY;
  host           varchar2(64) := 'smtp.email.[region].oci.oraclecloud.com';
  port           pls_integer := 25;
  sender         varchar2(64) := '[sender-address@domain.com]';
  recipient      varchar2(64) := '[recipient-address@domain.com]';
BEGIN
  select property_value into m_wallet_path
    from database_properties
   where property_name like '%SSL%';
  m_wallet_path := 'file:' || m_wallet_path;

  -- secure connection with SSL/TLS made *before* SMTP communication
  -- use hostname matched against CN or SMTP server's certificate
  reply := UTL_SMTP.OPEN_CONNECTION (
    host           => host,
    port           => port,
    c              => c,
    wallet_path    => m_wallet_path,
    wallet_password => NULL,
    secure_connection_before_smtp => FALSE,
    secure_host    => NULL);
  if (reply.CODE != 220) then
    raise_application_error(-20000, 'UTL_SMTP.OPEN_CONNECTION: '
      || reply.CODE || ' - ' || reply.TEXT);
  end if;

  UTL_SMTP.STARTTLS(c);

  -- username is the 'SMTP credential' for the user's email address
  -- obtained from OCI Identity -> Users -> email address.
  -- Scheme refers to SASL (Simple Authentication and Security Layer)
  -- Authentication Schemes for SMTP protocol - PLAIN, CRAM-MD5, LOGIN
  -- https://docs.oracle.com/en-us/iaas/Content/Email/Concepts/overview.htm
  UTL_SMTP.AUTH(
    c           => c,
    username    => '[ocid1.user.verylongaddresscreatedinOCIforuseraddress.com]',
    password    => '[password-provided-from-create]',

```

```

schemes => 'PLAIN');

UTL_SMTP.HELO(c, host);
UTL_SMTP.MAIL(c, sender);
UTL_SMTP.RCPT(c, recipient);
UTL_SMTP.OPEN_DATA(c);
UTL_SMTP.WRITE_DATA(c, 'From: Sender <' || sender || '>' || UTL_TCP.CRLF);
UTL_SMTP.WRITE_DATA(c, 'To: Recipient <' || recipient || '>' || UTL_TCP.CRLF);
UTL_SMTP.WRITE_DATA(c, 'Subject: Email from UTL_SMTP App' || UTL_TCP.CRLF);
UTL_SMTP.WRITE_DATA(c, 'Mail Content');
UTL_SMTP.WRITE_DATA(c, 'Sender is a domain-owned approved sender in OCI Mail Service');
UTL_SMTP.CLOSE_DATA(c);
dbms_output.put_line('Close Data: ' || reply.code);
UTL_SMTP.QUIT(c);

END;
/

```

Send Messages as Email from Autonomous AI Database

APPLIES TO:  Oracle Public Cloud only

You can use the DBMS_CLOUD_NOTIFICATION to send messages as an email.

1. Identify your SMTP connection endpoint for Email Delivery. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region.

For example, select one of the following for the SMTP connection endpoint:

- smtp.us-phoenix-1.oraclecloud.com
- smtp.us-ashburn-1.oraclecloud.com
- smtp.email.uk-london-1.oci.oraclecloud.com
- smtp.email.eu-frankfurt-1.oci.oraclecloud.com

See [Configure SMTP Connection](#) for more information.

2. Generate SMTP credentials for Email Delivery. UTL_SMTP uses credentials to authenticate with Email Delivery servers when you send email.

See [Generate SMTP Credentials for a User](#) for more information.

3. Create an approved sender for Email Delivery. Complete this step for all email addresses you use as the "From" with UTL_SMTP.MAIL.

See [Managing Approved Senders](#) for more information.

4. Allow SMTP access for ADMIN user by appending an Access Control Entry (ACE).

For example:

```

BEGIN
  -- Allow SMTP access for user ADMIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'www.us.example.com',
    lower_port => 587,
    upper_port => 587,
    ace => xs$acl_type(privilege_list => xs$name_list('SMTP'),
                      principal_name => 'ADMIN',
                      principal_type => xs_acl.ptype_db));

```

```
END;
/
```

5. Create a credential object and use `DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE` to send a message as an email.

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

BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
    provider          => 'email',
    credential_name   => 'EMAIL_CRED',
    message           => 'Subject content',
    params            => json_object('recipient' value
'mark@example.com, suresh@example.com',
                                'to_cc' value 'nicole@example.com,
jordan@example.com',
                                'to_bcc' value
'manisha@example.com',
                                'subject' value 'Test subject',
                                'smtp_host' value
'smtp.email.example.com',
                                'sender' value
'approver_sender@example.com' )
  );
END;
/
```

Use the `params` parameter to specify the sender, `smtp_host`, subject, recipient, and recipients of a CC or BCC in string values.

- *sender*: specifies the Email ID of the approved sender from Step 3.
- *smtp_host*: specifies the SMTP host name from step 2.
- *subject*: specifies the subject of the email.
- *recipient*: specifies the email IDs of recipients. Use a comma between email IDs when there are multiple recipients.
- *to_cc*: specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients.
- *to_bcc*: specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients.

See `SEND_MESSAGE` Procedure for more information.

Send Query Results as Email from Autonomous AI Database

APPLIES TO:  Oracle Public Cloud only

You can use the DBMS_CLOUD_NOTIFICATION package to send the results of a query as an email.

To use DBMS_CLOUD_NOTIFICATION to send mail:

1. Identify your SMTP connection endpoint for Email Delivery. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region.

For example, select one of the following for the SMTP connection endpoint:

- smtp.us-phoenix-1.oraclecloud.com
- smtp.us-ashburn-1.oraclecloud.com
- smtp.email.uk-london-1.oci.oraclecloud.com
- smtp.email.eu-frankfurt-1.oci.oraclecloud.com

See [Configure SMTP Connection](#) for more information.

2. Generate SMTP credentials for Email Delivery. UTL_SMTP uses credentials to authenticate with Email Delivery servers when you send email.

See [Generate SMTP Credentials for a User](#) for more information.

3. Create an approved sender for Email Delivery. Complete this step for all email addresses you use as the "From" with UTL_SMTP.MAIL.

See [Managing Approved Senders](#) for more information.

4. Allow SMTP access for ADMIN user by appending an Access Control Entry (ACE).

For example:

```
BEGIN
  -- Allow SMTP access for user ADMIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'www.us.example.com',
    lower_port => 587,
    upper_port => 587,
    ace => xs$ace_type(privilege_list => xs$name_list('SMTP'),
                      principal_name => 'ADMIN',
                      principal_type => xs_acl.ptype_db));
END;
/
```

5. Create a credential object and use DBMS_CLOUD_NOTIFICATION.SEND_DATA to send the output of a query as an email.

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

BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_DATA(
    provider        => 'email',
    credential_name => 'EMAIL_CRED',
```



```

        query          => 'SELECT tablespace_name FROM dba_tablespace',
        params         => json_object('recipient' value
'mark@example.com, suresh@example.com',
                                'to_cc' value
'mnicole@example.com1, jordan@example.com',
                                'to_bcc' value
'manisha@example.com',
                                'subject' value 'Test subject',
                                'type' value 'json',
                                'title' value 'mytitle',
                                'message' value 'This is the
message',
                                'smtp_host' value
'smtp.email.example.com',
                                'sender' value
'approver_sender@example.com' )
    );
END;
/

```

Use the `params` parameter to specify the sender, `smtp_host`, subject, recipient, recipients of a CC or BCC, the message, data type, and the title in String values.

- *sender*: specifies the Email ID of the approved sender from Step 3.
- *smtp_host*: specifies the SMTP host name from step 2.
- *subject*: specifies the subject of the email. The maximum size is 100 characters.
- *recipient*: This specifies the email IDs of recipients. Use a comma between email IDs when there are multiple recipients.
- *to_cc*: specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients.
- *to_bcc*: specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients.
- *message*: specifies the message text.
- *type*: specifies the output format as either CSV or JSON.
- *title*: specifies the title of the attachment of SQL output. The title should only contain letters, digits, underscores, hyphens, or dots as characters in its value due to it being used to generate a file name.

The maximum message size for use with `DBMS_CLOUD_NOTIFICATION.SEND_DATA` for mail notification is 32k bytes.

See `SEND_DATA` Procedure for more information.

Send Microsoft Teams Notifications from Autonomous AI Database on Dedicated Exadata Infrastructure

You can configure Microsoft Teams to send messages, alerts, or output of a query from Autonomous AI Database on Dedicated Exadata Infrastructure to a Microsoft Teams channel using the `DBMS_CLOUD_NOTIFICATION` package.

Prepare to Send Microsoft Teams Notifications from Autonomous AI Database

Get started by configuring a bot in your Microsoft Teams app. Next, create a credential to use with the DBMS_CLOUD_NOTIFICATION procedures to send Microsoft Teams notifications from Autonomous AI Database.

To configure Microsoft Teams notifications:

1. Create your Microsoft Teams app and add a bot in it. See [Developer Portal for Teams](#) for information on setting up an app.
 2. From the **Bot Management** section, ensure that the bot has a secret key, scope set to *Team* and permission to send notifications.
 3. Publish the app to your org to make it available to the people in your org.
 4. After your IT admin approves the app from the admin center, install the app from the **Apps** section in Teams.
 5. Request the Files.ReadWrite.All and ChannelSettings.Read.All permissions to the app for Graph API from the Azure Portal using the following instructions:
 - a. Log into your Azure Portal, navigate to **Azure Active Directory** using the left panel, and select the **Application** option.
 - b. The Application page displays the apps that you own along with the bots added to those apps. Click the bot to view its details.
 - c. Copy the *directory/tenant id* from the bot overview page for later use.
 - d. Then, go to **API permissions** in the left panel. Under **API permissions**, click **Add permission**, select **Microsoft graph** and then **Application permission**.
 - e. Search for Files.ReadWrite.All and ChannelSettings.Read.All permissions and add them.
 6. Have your IT admin approve the above-requested permissions from the Azure portal by following the steps below:
 - a. Log into your Azure Portal, navigate to **Azure Active Directory** using the left panel, and select the **Application** option.
 - b. Select **All applications** from the **Application** page.
 - c. Search the *application/bot* by its name, go to **API permissions**, and grant the ADMIN consent for the requested permissions: Files.ReadWrite.All and ChannelSettings.Read.All.
-  **Tip**

After your app is approved by the IT Admin, you can provide the bot id and secret key to other users to install the app within Teams in the org.
7. After the app is approved by the IT admin and the above requested permissions are granted, you can use the app's bot ID and secret key are used to create the credential object and generate a bot token.
 8. To send a query result to a Microsoft Teams channel, obtain the team id and the tenant id.

 Tip

The `team_id` is located in the team link between `/team/` and `/conversations`. The `tenant_id` is located after `"tenantId="` at the end of the team link. This link is found by clicking the three dots next to the team name and selecting **Get link to team**.

For example:

```
https://teams.microsoft.com/l/team/teamID/conversations?
groupId=groupid%tenantId=tenantid
```

9. Obtain the `channelID`.

 Tip

`channelID` is located in the channel link between `/team/` and the channel name. This link is found by clicking the three dots next to the channel name and selecting **Get link to channel**.

For example:

```
https://teams.microsoft.com/l/channel/channelID/channel_name?
groupId=groupid&tenantId=tenantid
```

10. On Exadata Cloud@Customer, configure the HTTP Proxy settings in your environment to allow the database to access Microsoft Teams. See *Using the Console to Provision Exadata Database Service on Cloud@Customer* for information on setting up the HTTP Proxy settings while creating an Exadata Cloud@Customer Infrastructure.

 Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in *Requires Activation* state. Once it is activated, you cannot edit those settings.

To set up an HTTP Proxy for an already provisioned Exadata Infrastructure, please submit a Service Request (SR) in My Oracle Support. See *Create a Service Request in My Oracle Support* for details.

11. Create a credential object to access the Microsoft Teams app from Autonomous AI Database.

 Tip

If you can not use the `CREATE_CREDENTIAL` procedure successfully, consult with the ADMIN user to grant execute access on `DBMS_CLOUD` packages.

The credential's username is the `bot_id` and the password is the bot key.

For example:

```
BEGIN
```

```

        DBMS_CLOUD.CREATE_CREDENTIAL(credential_name => 'TEAMS_CRED',
        username          => 'bot_id',
        password          => 'bot_secret');
    END;
/

```

See CREATE_CREDENTIAL Procedure for more information.

Send Messages to a Microsoft Teams Channel

After creating the Microsoft Teams credential object as described in [Prepare to Send Microsoft Teams Notifications from Autonomous AI Database](#), you can use the DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE procedure to send a message to a Microsoft Teams channel.

Example:

```

BEGIN
    DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider          => 'msteams',
        credential_name   => 'TEAMS_CRED',
        message           => 'text from new teams api',
        params            => json_object('channel' value 'channelID'));
END;
/

```

Use the params parameter to specify the channel.

- *channel*: specifies the Channel ID obtained from Step 10 in [Prepare to Send Microsoft Teams Notifications from Autonomous AI Database](#).

See SEND_MESSAGE Procedure for more information.

Send Query Results to a Microsoft Teams Channel

After creating the Microsoft Teams credential object as described in [Prepare to Send Microsoft Teams Notifications from Autonomous AI Database](#), you can use the DBMS_CLOUD_NOTIFICATION.SEND_DATA procedure to send the output of a query to a Microsoft Teams channel.

Example:

```

BEGIN
    DBMS_CLOUD_NOTIFICATION.SEND_DATA(provider => 'msteams',
        credential_name => 'TEAMS_CRED',
        query           => 'SELECT tablespace_name FROM dba_tablespace',
        params          => json_object('tenant' value '5b743bc*****c0286',
        'team' value '0ae401*****5d2bd',
        'channel' value
        '19%3a94be023*****%40thread.tacv2',
        'title' value 'today',
        'type' value 'csv'));
END;
/

```

Use the `params` parameter to specify the tenant, team, channel, title, and data type in string values.

- *tenant*: specifies the tenant ID obtained from Step 8 in [Prepare to Send Microsoft Teams Notifications from Autonomous AI Database](#).
- *team*: specifies the team ID obtained from Step 8 in [Prepare to Send Microsoft Teams Notifications from Autonomous AI Database](#).
- *channel*: specifies the channel ID obtained from Step 9 in [Prepare to Send Microsoft Teams Notifications from Autonomous AI Database](#).
- *title*: specifies the title of the file. The title can only contain alphabets, digits, underscores, and hyphens. The file name that appears in Microsoft Teams will be a concatenation of the title parameter and the timestamp to ensure uniqueness. The maximum title size is 50 characters.

For example: `'title'_'timestamp'_'format'`

- *type*: This specifies the output format. Valid values are CSV or JSON.

 Note

The maximum file size supported when using `DBMS_CLOUD_NOTIFICATION.SEND_DATA` for Microsoft Teams is 4MB.

See `SEND_DATA` Procedure for more information.

Send Slack Notifications from Autonomous AI Database on Dedicated Exadata Infrastructure

You can configure Slack to send messages, alerts, or output of a query from Autonomous AI Database on Dedicated Exadata Infrastructure to a Slack channel using the `DBMS_CLOUD_NOTIFICATION` package.

Prepare to Send Slack Notifications from Autonomous AI Database

To send Slack notifications, you must configure your Slack application to receive messages from Autonomous AI Database. Next, create a credential to use with the `DBMS_CLOUD_NOTIFICATION` procedures to send Slack notifications from Autonomous AI Database.

To use Slack with `DBMS_CLOUD_NOTIFICATION` procedures:

1. Create your Slack app.

The Slack app is installed in a Slack Workspace, which in turn has Channels where messages can be sent. The bot token of the Slack app must have the following permission scopes defined:

```
channels:read
chat:write
files:write
```

See [Creating an app](#) for information on setting up a Slack app.

2. Have your Slack admin add the Slack app to the channels to which DBMS_CLOUD_NOTIFICATION can send message through the "Integrations" option in the channel.

See [Basic app setup](#) for more information.

3. Locate the app's bot token available in the Slack app specific page, under <https://app.slack.com>.

See [Basic app setup](#) for more information.

4. On Exadata Cloud@Customer, configure the HTTP Proxy settings in your environment to allow the database to access Slack. See [Using the Console to Provision Exadata Database Service on Cloud@Customer](#) for information on setting up the HTTP Proxy settings while creating an Exadata Cloud@Customer Infrastructure.

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in Requires Activation state. Once it is activated, you cannot edit those settings.

To set up an HTTP Proxy for an already provisioned Exadata Infrastructure, please submit a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

5. Create a credential object to access the Slack app from Autonomous AI Database.

Tip

If you can not use the CREATE_CREDENTIAL procedure successfully, consult with the ADMIN user to grant execute access on DBMS_CLOUD packages.

The credential's username is SLACK_TOKEN and the password is the bot token.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'SLACK_CRED',
    username       => 'SLACK_TOKEN',
    password       => 'xoxb-34....96-34....52-zw....cy');
END;
/
```

See CREATE_CREDENTIAL Procedure for more information.

6. Configure access control to allow user access to external network services (Slack).

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
    host          => 'slack.com',
    lower_port    => 443,
    upper_port    => 443,
    ace           => xs$ace_type(
      privilege_list => xs$name_list('http'),
```

```

        principal_name => example_invoking_user,
        principal_type => xs_acl.ptype_db));
END;
```

7. If you are sending SQL, configure access control to allow sending of output with `DBMS_CLOUD_NOTIFICATION.SEND_DATA`.

```

BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
    host          => 'slack.com',
    lower_port    => 443,
    upper_port    => 443,
    ace           => xs$ace_type(
      privilege_list => xs$name_list('http'),
      principal_name => example_invoking_user,
      principal_type => xs_acl.ptype_db));
END;
```

See *Configuring Access Control for External Network Services* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for more information.

Send Messages to a Slack Channel

After creating the Slack credential object as described in [Prepare to Send Slack Notifications from Autonomous AI Database](#), you can use the `DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE` procedure to send a message to a Slack channel.

Example:

```

BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
    provider          => 'slack',
    credential_name   => 'SLACK_CRED',
    message           => 'Alert from Autonomous Database...',
    params            => json_object('channel' value 'C0....08'));
END;
```

Use the `params` parameter to specify the Slack channel.

- `channel`: Specifies the Channel ID.

The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab. See [How to Find your Slack Team ID and Slack Channel ID](#) for more information.

See `SEND_MESSAGE` Procedure for more information.

Send Query Results to a Slack Channel

After creating the Slack credential object as described in [Prepare to Send Slack Notifications from Autonomous AI Database](#), you can use the `DBMS_CLOUD_NOTIFICATION.SEND_DATA` procedure to send the output of a query to a Slack channel.

Example:

```
BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_DATA(
    provider => 'slack',
    credential_name => 'SLACK_CRED',
    query => 'SELECT username, account_status, expiry_date FROM USER_USERS
WHERE rownum < 5',
    params => json_object('channel' value 'C0...08',
                          'type' value 'csv'));
END;
/
```

Use the `params` parameter to specify the Slack channel and the data type:

- `channel`: Specifies the Channel ID.
The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab. See [How to Find your Slack Team ID and Slack Channel ID](#) for more information.
- `type`: Specifies the output type. Valid values are: 'csv' or 'json'.

See `SEND_DATA` Procedure for more information.

Make External Calls using a Customer-Managed Wallet

You can use the `UTL_HTTP` package when your Autonomous AI Database needs to access data on the internet over HTTP/HTTPS. The `UTL_HTTP` package lets you make HTTP callouts directly from SQL and PL/SQL.

If you're calling an **HTTPS** endpoint, you must configure an **Oracle Wallet**. Autonomous AI Database requires a wallet that contains the trusted root and intermediate certificates for any HTTPS endpoint your database will connect to. The `UTL_HTTP` package uses this wallet to establish a secure SSL/TLS connection. You can create and manage the wallet with the `orapki` utility.

Note

Plain HTTP (non-HTTPS) requests don't require an Oracle Wallet.

The sections below explain how to configure and use a customer-managed wallet to make outbound HTTPS calls with the `UTL_HTTP` package on Autonomous AI Database.

Prerequisites to Use a Customer-Managed Wallet with External Calls

Before you start, make sure you have:

- Access to the Autonomous AI Database using an account that can run PL/SQL and configure `UTL_HTTP`.
- Access to the target HTTPS endpoint and its full certificate chain (root + intermediate certificates).
- A workstation where you can run **orapki** to create and validate an Oracle Wallet. You need to install Oracle Database to get `orapki` utility. See [Installing Oracle Database](#) for details.

Prepare the Customer-Managed Wallet

In this step, you'll create and validate the wallet on your workstation before uploading it to Autonomous AI Database.

Obtain or create a customer-managed wallet

Example using orapki:

```
-- Create an SSL Wallet and load the Root CERTs using orapki utility
$ORACLE_HOME/bin/orapki wallet create -wallet /u01/web/wallet -pwd *****
$ORACLE_HOME/bin/orapki wallet add -wallet /u01/web/wallet -trusted_cert -
cert MyWebServer.cer -pwd *****
-- Store the credentials in the SSL Wallet using mkstore utility
$ORACLE_HOME/bin/mkstore -wrl /u01/web/wallet -createCredential secret-from-
the-wallet 'example@oracle.com'
*****Enter wallet password: *****
```

Validate the wallet

```
$ORACLE_HOME/bin/orapki wallet display -wallet /u01/web/wallet
```

You should see all imported certificates listed.

Upload the wallet

Once the customer-managed wallet is ready (including any required self-signed/root/intermediate certificates), upload the wallet files to a location in Oracle Cloud Infrastructure (OCI) Object Storage.

Use a customer-managed wallet with UTL_HTTP

This section shows how to download your wallet from Object Storage, allow network access to the HTTPS endpoint, and then make HTTPS calls using the UTL_HTTP package.

1. Create a credential for Object Storage access:

```
BEGIN
DBMS_CLOUD.CREATE_CREDENTIAL(credential_name => 'DEF_CRED_NAME',
username => 'user1@example.com',
password => 'password'
);
END;
/
```

The values you provide for username and password depend on the Cloud Object Storage service you are using. This creates the credential you use to access the Cloud Object Storage where the customer-managed wallet resides.

2. Create (or reuse) a directory object for the wallet:

Use an existing directory or create a new a directory for the wallet file. For example:

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

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

3. Get the absolute directory path:

You'll need the absolute directory path when calling UTL_HTTP.SET_WALLET.

```
SELECT DIRECTORY_NAME, DIRECTORY_PATH FROM DBA_DIRECTORIES WHERE
DIRECTORY_NAME = '<wallet_dir>';
```

4. Download the wallet file from Object Storage into the directory:

Use DBMS_CLOUD.GET_OBJECT to copy the wallet file into your directory. For example:

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

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

5. Allow outbound network access (ACL) to the HTTPS endpoint:

You must allow the database user/schema to reach the target host over the network.

```
BEGIN
DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
host => 'api.example.com',
ace => xs$ace_type(
privilege_list => xs$name_list('CONNECT','HTTP','RESOLVE'),
principal_name => 'ADMIN',
principal_type => xs_acl.ptype_db
)
);
END;
/
```

On Exadata Cloud@Customer deployments, you also need to setup proxy as shown below:

```
BEGIN
UTL_HTTP.SET_PROXY('www-proxy.us.oracle.com:80', 'oracle.com');
END;
/
```

6. Make HTTPS calls with UTL_HTTP:

Simple GET request :

```
DECLARE
l_http_req UTL_HTTP.req;
l_http_resp UTL_HTTP.resp;
```

```

l_response VARCHAR2(32767);

BEGIN
    utl_http.set_wallet('file:<absolute_directory_path_from_step_3>');
    l_http_req := UTL_HTTP.BEGIN_REQUEST('https://api.example.com/status');
    l_http_resp := UTL_HTTP.GET_RESPONSE(l_http_req);
    LOOP UTL_HTTP.READ_LINE(l_http_resp, l_response, TRUE);
        DBMS_OUTPUT.PUT_LINE(l_response);
    END LOOP;
    UTL_HTTP.END_RESPONSE(l_http_resp);
END;
/

```

POST request with JSON payload:

```

DECLARE
    l_req UTL_HTTP.req;
    l_resp UTL_HTTP.resp;
    l_line VARCHAR2(32767);
BEGIN
    utl_http.set_wallet('file:<absolute_directory_path_from_step_3>');
    l_req := UTL_HTTP.BEGIN_REQUEST( url => 'https://api.example.com/data',
method => 'POST' );
    UTL_HTTP.SET_HEADER(l_req, 'Content-Type', 'application/json');
    UTL_HTTP.WRITE_TEXT(l_req, '{"key":"value"}');
    l_resp := UTL_HTTP.GET_RESPONSE(l_req);
    LOOP UTL_HTTP.READ_LINE(l_resp, l_line, TRUE);
        DBMS_OUTPUT.PUT_LINE(l_line);
    END LOOP;
    UTL_HTTP.END_RESPONSE(l_resp);
END;
/

```

Troubleshooting for common errors:

Certificate chain errors

Error: ORA-29024: Certificate validation failure

- Root and intermediate certificates may be missing.
- The target endpoint may be using a new CA - re-download full cert chain and re-create the wallet.

Wallet path errors

ORA-28759: Failure to open file

- Incorrect wallet path in SET_WALLET.
- Missing file: prefix.
- Uploaded wallet files not present in directory.

Handshake failures

ORA-24263 / ORA-29005

- TLS protocol mismatch.

- Bad or expired certificates.
- Endpoint enforces TLS $\geq$ 1.2

Use a Customer-Managed Wallet for Scheduler Email Notifications (SMTP)

This section shows how to configure Scheduler email notifications to use an SMTP server over TLS (STARTTLS) with a customer-managed wallet.

Before you begin, make sure you've already prepared the customer-managed wallet (created it locally, validated it, and uploaded it to Object Storage). See [Prepare the Customer-Managed Wallet](#) for details.

To use a customer-managed wallet with the scheduler email server:

1. Create a credential for Object Storage access:

You can use `DBMS_CLOUD.CREATE_CREDENTIAL` to create a credential to access Cloud Object Storage.

```
BEGIN
DBMS_CLOUD.CREATE_CREDENTIAL(
  credential_name => 'DEF_CRED_NAME',
  username => 'user1@example.com',
  password => 'password'
);
END;
/
```

The values you provide for username and password depend on the Cloud Object Storage service you are using. This creates the credential you use to access the Cloud Object Storage where the customer-managed wallet resides.

2. Create (or reuse) a directory object for the wallet:

Use an existing directory or create a new a directory for the wallet file. For example:

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

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

3. Get the absolute directory path:

You'll need the absolute directory path when calling `UTL_HTTP.SET_WALLET`.

```
SELECT DIRECTORY_NAME, DIRECTORY_PATH FROM DBA_DIRECTORIES WHERE
DIRECTORY_NAME = '<wallet_dir>';
```

4. Download the wallet file from Object Storage into the directory:

Use `DBMS_CLOUD.GET_OBJECT` to copy the wallet file into your directory. For example:

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

```
END;
/
```

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

5. Configure Scheduler email (SMTP + STARTTLS):

Run the commands to set up the scheduler to send SMTP email for scheduler job notifications:

```
EXEC DBMS_CLOUD.CREATE_CREDENTIAL('EMAIL_CRED', '<user_ocid>',
'<password>');
GRANT EXECUTE ON admin.EMAIL_CRED TO sys;
EXEC DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    'EMAIL_SERVER',
    'smtp.email.us-ashburn-1.oci.oraclecloud.com:587'
);
EXEC DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    'EMAIL_SERVER_CREDENTIAL',
    'EMAIL_CRED'
);
EXEC DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    'EMAIL_SERVER_ENCRYPTION',
    'STARTTLS'
);
```

The commands does the following:

- Sets the Scheduler SMTP endpoint (EMAIL_SERVER)
- Stores SMTP login details in EMAIL_CRED and points Scheduler to it
- Enables TLS encryption using STARTTLS

See [SET_SCHEDULER_ATTRIBUTE Procedure](#) for more information.

6. Create a credential for the wallet password:

Create a credential to store the password for the customer-managed wallet.

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'WALLET_CRED',
        username         => 'any_user',
        password          => 'password'
    );
END;
/
```

This creates the credential you use in the next step to provide the password for the customer-managed wallet.

7. Tell Scheduler where the wallet is and how to unlock it:

Set the scheduler wallet directory and the wallet credential.

```
BEGIN
  DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    'EMAIL_SERVER_WALLET_DIRECTORY',
    'WALLET_DIR'
  );

  DBMS_SCHEDULER.SET_SCHEDULER_ATTRIBUTE(
    'EMAIL_SERVER_WALLET_CREDENTIAL',
    'ADMIN.WALLET_CRED'
  );
END;
/
```

8. Verify the Scheduler email configuration:

Query the DBA_SCHEDULER_GLOBAL_ATTRIBUTE view to verify the values you set in the previous steps.

```
SELECT attribute_name, value
FROM DBA_SCHEDULER_GLOBAL_ATTRIBUTE
WHERE attribute_name LIKE 'EMAIL_SERVER%' ORDER BY 1, 2;
ATTRIBUTE_NAME          VALUE
-----
EMAIL_SERVER             smtp.email.us-
ashburn-1.oci.oraclecloud.com:587
EMAIL_SERVER_CREDENTIAL  "ADMIN"."EMAIL_CRED"
EMAIL_SERVER_ENCRYPTION  STARTTLS
EMAIL_SERVER_WALLET_CREDENTIAL "ADMIN"."WALLET_CRED"
EMAIL_SERVER_WALLET_DIRECTORY "WALLET_DIR"
```

Reference:

Useful orapki commands:

```
orapki wallet display -wallet <path> orapki wallet add -wallet <path>
-trusted_cert -cert <cert-file> orapki wallet create -wallet <path> -
auto_login
```

Sample wallet directory structure:

```
cmw_wallet/
├── ewallet.p12
└── cwallet.sso
```

Manage and Store Files in a Cloud Code Repository with Autonomous AI Database

Autonomous AI Database provides routines to manage and store files in Cloud Code (Git) Repositories. The supported Cloud Code Repositories are: GitHub, AWS CodeCommit, and Azure Repos.

About Cloud Code Repositories with Autonomous AI Database

The DBMS_CLOUD_REPO package provides a single interface for accessing a Cloud Code Repository from Autonomous AI Database.

The supported Cloud Code Repositories provide the following features:

- **Git Version Control System:** [Git](#) is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
- **Git Repository:** A Git repository is a virtual storage of your project. It allows you to save versions of your code, which you can access when needed.

The DBMS_CLOUD_REPO APIs use a repository handle (REPO object). The repository handle is an opaque JSON object that represents a Cloud Code Repository of a specific cloud provider. A REPO object can be passed to different DBMS_CLOUD_REPO APIs. This opaque object ensures that DBMS_CLOUD_REPO procedures and functions are multicloud compatible; you do not have to change your code when you migrate from one Cloud Code Repository provider to another Cloud Code Repository.

Autonomous AI Database provides the following to help you work with Cloud Code Repositories:

- Repository initialization operations to allow you to initialize a repository.
See [Initialize a Cloud Code Repository](#) for more information.
- Repository management operations that let you create, list, update or delete a repository.
See [Create and Manage a Cloud Code Repository](#) for more information.
- Repository branch management operations that let you create, list, merge or delete branches in a repository.
See [Create and Manage Branches in a Cloud Code Repository](#) for more information.
- Export the metadata DDL of all objects in a schema to a repository.
See [Export Schema Objects to the Cloud Code Repository Branch](#) for more information.
- Repository file management operations to upload, download, update, and delete files.
See [Use File Operations with a Cloud Code Repository](#) for more information.
- SQL install operations that let you export database object metadata DDL to a repository and install SQL statements into the database from a Cloud Code Repository.
See [Use SQL Install Operations with a Cloud Code Repository](#) for more information.

Initialize a Cloud Code Repository

The DBMS_CLOUD_REPO initialization routines initialize a Cloud Code Repository. After you obtain a Cloud Code Repository handle, you use the handle to access the Cloud Code Repository.

To initialize a Cloud Code Repository:

1. Create a credential to access the Cloud Code Repository.

See CREATE_CREDENTIAL Procedure for information on creating credentials.

2. Depending on the repository, GitHub, Azure Repos, or AWS CodeCommit, call `DBMS_CLOUD_REPO.INIT_REPO` with the parameters for the particular repository to obtain a repository handle.

The following examples provide samples for each supported Cloud Code Repository.

- **GitHub Initialization:**

```

DEFINE repo_name='test_repo';
DEFINE cred_name='GITHUB_CRED';
VAR repo clob
BEGIN
    :repo := DBMS_CLOUD_REPO.INIT_REPO(
        params => JSON_OBJECT('provider' value 'github',
                              'repo_name' value '&repo_name',
                              'credential_name' value
                              '&cred_name',
                              'owner' value
                              '<myuser>')
    );
END;
/

```

- **Azure Repos Initialization:**

```

DEFINE repo_name='test_repo';
DEFINE cred_name='AZURE_REPO_CRED';
VAR repo clob
BEGIN
    :repo := DBMS_CLOUD_REPO.INIT_REPO(
        params => JSON_OBJECT('provider' value 'azure',
                              'repo_name' value '&repo_name',
                              'credential_name' value
                              '&cred_name',
                              'organization' value '<myorg>',
                              'project' value '<myproject>')
    );
END;
/

```

- **AWS CodeCommit Initialization:**

```

DEFINE repo_name='test_repo';
DEFINE cred_name='AWS_REPO_CRED';
VAR repo clob
BEGIN
    :repo := DBMS_CLOUD_REPO.INIT_REPO(
        params => JSON_OBJECT('provider' value 'aws',
                              'repo_name' value '&repo_name',
                              'credential_name' value
                              '&cred_name',
                              'region' value 'us-east-1')
    );
END;
/

```

See `DBMS_CLOUD_REPO` Initialization Operations for details on the initialization functions.

Create and Manage a Cloud Code Repository

The DBMS_CLOUD_REPO management routines allow you to manage a Cloud Code Repository by creating, listing, updating, or deleting a repository.

First, obtain a Cloud Code Repository handle to provide access a repository. See [Initialize a Cloud Code Repository](#) for details.

1. To create a repository:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.CREATE_REPOSITORY(
    repo => :repo,
    description => 'test repo'
  );
END;
/
```

2. To update a repository:

```
VAR repo clob
DEFINE repo_name='test_repo';
BEGIN
  DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
    repo => :repo,
    new_name => '&repo_name' || '_new'
  );
END;
/
```

3. To list repositories:

```
col id format a30
col name format a10
col description format a15
select id, name, bytes, private, description from
  DBMS_CLOUD_REPO.LIST_REPOSITORIES(:repo);
```

4. To delete a repository:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.DELETE_REPOSITORY(
    repo => :repo
  );
END;
/
```

See DBMS_CLOUD_REPO Repository Management Operations for more information.

Create and Manage Branches in a Cloud Code Repository

The DBMS_CLOUD_REPO management routines allow you to manage Cloud Code Repository branches by creating, listing, merging, or deleting branches in a repository.

To perform Cloud Code Repository branch management operations you must first:

- Create a credential.
See [CREATE_CREDENTIAL Procedure](#) for details.
 - Obtain a handle.
See [Initialize a Cloud Code Repository](#) for details.
 - Create a repository.
See [Create and Manage a Cloud Code Repository](#) for details.
 - Log in as the ADMIN user or have EXECUTE privilege on DBMS_CLOUD_REPO.
1. To create a branch in a Cloud Code Repository:

```
BEGIN
  DBMS_CLOUD_REPO.CREATE_BRANCH (
    repo          => l_repo,
    branch_name    => 'test_branch',
    parent_branch  => 'main'
  );
END;
/
```

2. To delete a branch in a Cloud Code Repository:

```
BEGIN
  DBMS_CLOUD_REPO.DELETE_BRANCH (
    repo          => l_repo,
    branch_name    => 'test_branch'
  );
END;
/
```

3. To merge a branch into another branch in a Cloud Code Repository:

```
BEGIN
  DBMS_CLOUD_REPO.MERGE_BRANCH (
    repo          => l_repo,
    branch_name    => 'test_branch',
    target_branch_name => 'main'
  );
END;
/
```

4. To list branches in a Cloud Code Repository:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_BRANCHES (repo => l_repo);
```

5. You can list commits in a branch in Cloud Code Repository based on the following;
- Repository
 - Branch
 - The file path of the branch
 - SHA/commit_id

Note

SHA is a checksum 40-character string composed of hexadecimal characters (0–9 and a–f).

To list commits based on the repository:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (  
    repo => :repo);
```

To list commits based on the repository, commit_id and file path of the branch:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (  
    repo      => l_repo,  
    commit_id => '66dd2b23b74cd0afabd11af66c6aa9c550540ba6',  
    file_path  => 'sub_dir/test11.sql')
```

To list commits based on the repository, and file path of the branch:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (  
    repo      => :repo,  
    branch_name => 'branch1');
```

To list commits based on the repository, branch_name and file path of the branch:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (  
    repo      => :repo,  
    branch_name => 'branch1',  
    file_path  => 'sub_dir/test11.sql');
```

To list commits based on the repository and commit_id of the branch:

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_COMMITS (  
    repo => :repo,  
    commit_id => '66dd2b23b74cd0afabd11af66c6aa9c550540ba6');
```

See DBMS_CLOUD_REPO Repository Branch Management Operations for more information.

Export Schema Objects to the Cloud Code Repository Branch

The DBMS_CLOUD_REPO management routine allows you to export metadata of the objects in a schema to the Cloud Code Repository branch. You can filter your list based on the object names or object types.

To export schema metadata you must first:

- Create a credential.
See [CREATE_CREDENTIAL Procedure](#) for details.
- Obtain a handle.
See [Initialize a Cloud Code Repository](#) for details.
- Create a repository.
See [Create and Manage a Cloud Code Repository](#) for details.
- Log in as the ADMIN user or have EXECUTE privilege on DBMS_CLOUD_REPO.

Use the EXPORT_SCHEMA procedure to export metadata of the objects in your schema to a Cloud Code Repository branch:

```
BEGIN
  DBMS_CLOUD_REPO.EXPORT_SCHEMA(
    repo      => l_repo,
    schema_name => 'USER1',
    file_path  => 'myschema_ddl.sql'
    filter_list =>
      to_clob('[
        { "match_type": "equal",
          "type": "table"
        },
        { "match_type": "not_equal",
          "type": "view"
        },
        { "match_type": "in",
          "type": "table",
          "name": " 'EMPLOYEE_SALARY' ', 'EMPLOYEE_ADDRESS' ' "
        },
        { "match_type": "equal",
          "type": "sequence",
          "name": "EMPLOYEE_RECORD_SEQ"
        },
        { "match_type": "like",
          "type": "table",
          "name": "%OFFICE%"
        }
      ]'
    );
END;
```

This example exports the metadata of the USER1 schema into the l_repo repository. The export includes the metadata of the tables EMPLOYEE_SALARY and EMPLOYEE_ADDRESS, and any table

name that contains OFFICE. It also exports the EMPLOYEE_RECORD_SEQ sequence and excludes the views in the schema.

See EXPORT_SCHEMA Procedure for more information.

Use File Operations with a Cloud Code Repository

The DBMS_CLOUD_REPO file operations allow you to create, get, list, update, or delete files in a Cloud Code Repository.

Obtain a Cloud Code Repository handle before using the file operations. See [Initialize a Cloud Code Repository](#) for details.

You also need to create a repository before you work with files. See [Create and Manage a Cloud Code Repository](#) for details.

1. To get a file:

```
SELECT DBMS_CLOUD_REPO.GET_FILE(repo => :repo, file_path => 'test1.sql')
```

2. To create a file:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.PUT_FILE(
    repo => :repo,
    file_path => 'test1.sql',
    contents => UTL_RAW.cast_to_raw('create table t1 (x varchar2(30))'
  || CHR(10) || '/')
  );
END;
/
```

3. To update a file:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.PUT_FILE(
    repo => :repo,
    file_path => 'test1.sql',
    contents => UTL_RAW.cast_to_raw('create table t2 (x varchar2(30))'
  || CHR(10) || '/')
  );
END;
/
```

4. To list files:

```
SELECT id, name, bytes, url FROM DBMS_CLOUD_REPO.LIST_FILES(repo => :repo);
```

5. To delete a file:

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.DELETE_FILE(
```

```

        repo => :repo,
        file_path => 'test1.sql'
    );
END;
/

```

See DBMS_CLOUD_REPO File Operations for more information.

Use SQL Install Operations with a Cloud Code Repository

The DBMS_CLOUD_REPO SQL Install operations allow you to store and download SQL scripts from a Cloud Code Repository.

Obtain a Cloud Code Repository handle before using the SQL Install operations. See [Initialize a Cloud Code Repository](#) for details.

You also need to create a repository before you work with SQL Install operations. See [Create and Manage a Cloud Code Repository](#) for details.

The scripts are intended as schema install scripts and not as generic SQL scripts:

- Scripts cannot contain SQL*Plus client specific commands.
- Scripts cannot contain bind variables or parameterized scripts.
- SQL statements must be terminated with a slash on a new line (/).
- Scripts can contain DDL, DML PLSQL statements, but direct SELECT statements are not supported. Using SELECT within a PL/SQL block is supported.

Any SQL statement that can be run using EXECUTE IMMEDIATE will work if it does not contain bind variables or defines.

1. To upload DDL metadata to a Cloud Code Repository:

```

VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.EXPORT_OBJECT(
        repo => :repo,
        object_type => 'PACKAGE',
        object_name => 'MYPACK',
        file_path => 'mypack.sql'
    );
END;
/

```

2. To install SQL statements from a file:

```

VAR repo clob
BEGIN
    DBMS_CLOUD_REPO.INSTALL_FILE(
        repo => :repo,
        file_path => 'test3.sql',
        stop_on_error => FALSE
    );
END;
/

```

3. To install SQL statements from a buffer::

```
VAR repo clob
BEGIN
  DBMS_CLOUD_REPO.INSTALL_SQL(
    repo => :repo,
    content => 'create table t1 (x varchar2(30))' || CHR(10) || '/',
    stop_on_error => FALSE
  );
END;
/
```

See DBMS_CLOUD_REPO SQL Install Operations for more information.

Cache Messages with Singleton Pipes

Singleton Pipe is an addition to the DBMS_PIPE package that allows you to cache and retrieve a custom message and share the message across multiple database sessions with concurrent reads.

About Caching Messages with Singleton Pipes

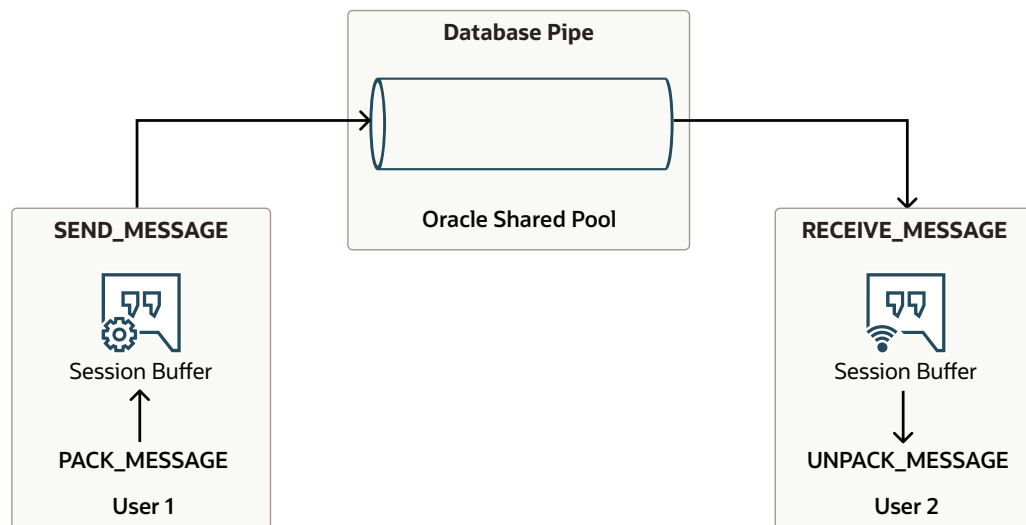
The DBMS_PIPE package has extended functionality on Autonomous AI Database to support Singleton Pipes.

A Singleton Pipe in DBMS_PIPE:

- Provides in-memory caching of custom data using Singleton Pipe messages.
- Supports the ability to cache and retrieve a custom message of up to 32,767 bytes.
- Supports sharing a cached message across multiple database sessions with concurrent reads. This provides high throughput and supports concurrent reads of messages across database sessions.
- Supports Read-Only and Read-Write databases.
- Supports several cache invalidation methods:
 - Explicit cache invalidation controlled by user.
 - Cache invalidation after a user specified time interval (in seconds). This invalidation method is controlled by the message sender, using the `shelfLife` parameter, instead of by message readers. This avoids the common pitfalls due to incorrect use of cache by readers.

About Standard Pipes and Singleton Pipes

The DBMS_PIPE Package allows two or more database sessions to communicate using in-memory messages. Pipe functionality has several applications such as external service interface, debugging, independent transactions, and alerts. See *DBMS_PIPE* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for more information.

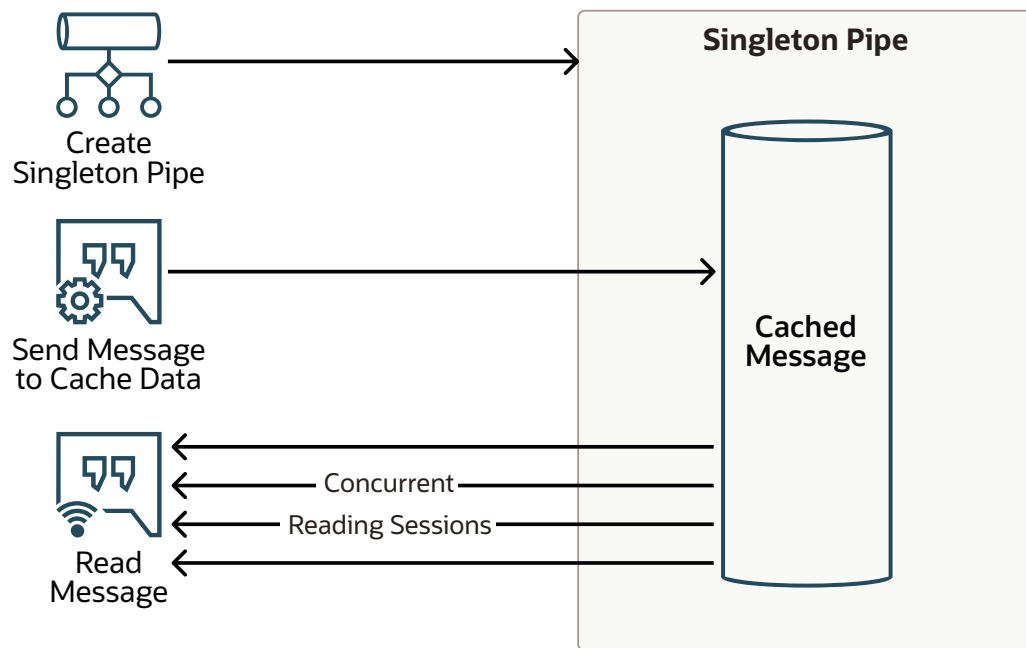


A Singleton Pipe can be any one of the supported DBMS_PIPE types:

- **Implicit Pipe:** Automatically created when a message is sent with an unknown pipe name using the `DBMS_PIPE.SEND_MESSAGE` function.
- **Explicit Pipe:** Created using the `DBMS_PIPE.CREATE_PIPE` function with a user specified pipe name.
- **Public Pipe:** Accessible by any user with `EXECUTE` permission on `DBMS_PIPE` package.
- **Private Pipe:** Accessible by sessions with the same user as the pipe creator.

Singleton Pipes provide the ability to cache a single message in the memory of the Autonomous AI Database instance.

The following shows the general workflow for using singleton pipes.



Singleton Pipe Overview and Features

- **Singleton Messages**
 - A Singleton Pipe can cache one message in the pipe, hence the name "singleton".

- The message in a Singleton Pipe can be comprised of multiple fields, up to a total message size of 32,767 bytes.
- DBMS_PIPE supports the ability to pack multiple attributes in a message using DBMS_PIPE.PACK_MESSAGE procedure.
- For a Public Singleton Pipe, the message can be received by any database session with execute privilege on DBMS_PIPE package.
- For Private Singleton Pipe, the message can be received by sessions with the same user as the creator of the Singleton Pipe.
- **High Message Throughput for Reads**
 - Singleton Pipes cache the message in the pipe until it is invalidated or purged. Database sessions can concurrently read a message from the Singleton Pipe.
 - Receiving a message from a Singleton Pipe is a non-blocking operation.
- **Message Caching**
 - A message is cached in a Singleton Pipe using DBMS_PIPE.SEND_MESSAGE.
 - If there is an existing cached message in the Singleton Pipe, then DBMS_PIPE.SEND_MESSAGE overwrites the previous message to maintain only one message in the Singleton Pipe.
- **Message Invalidation**
 - **Explicit Invalidation:** purges the pipe with the procedure DBMS_PIPE.PURGE or by overwriting the message using DBMS_PIPE.SEND_MESSAGE.
 - **Automatic Invalidation:** a message can be invalidated automatically after the specified shelflife time has elapsed.
- **No Eviction from Database Memory**
 - Singleton Pipes do not get evicted from Oracle Database memory.
 - An Explicit Singleton Pipe continues to reside in database memory until it is removed using DBMS_PIPE.REMOVE_PIPE or until the database restarts.
 - An Implicit Singleton Pipe stays in database memory until there is one cached message in the pipe.

Singleton Pipe Operations

Operation	DBMS_PIPE Function or Procedure
Create an Explicit Singleton Pipe	CREATE_PIPE Function
Cache a message in Singleton Pipe	<i>PACK_MESSAGE Procedures</i> in Oracle Database 19c PL/SQL Packages and Types Reference or Oracle Database 26ai PL/SQL Packages and Types Reference SEND_MESSAGE Function
Read a cached message from Singleton Pipe	RECIEVE_MESSAGE Function, <i>UNPACK_MESSAGE Procedures</i> in Oracle Database 19c PL/SQL Packages and Types Reference or Oracle Database 26ai PL/SQL Packages and Types Reference
Delete a message in Singleton Pipe	<i>PURGE Procedure</i> in Oracle Database 19c PL/SQL Packages and Types Reference or Oracle Database 26ai PL/SQL Packages and Types Reference

Operation	DBMS_PIPE Function or Procedure
Remove an Explicit Singleton Pipe	<i>REMOVE_PIPE</i> Function in Oracle Database 19c PL/SQL Packages and Types Reference or Oracle Database 26ai PL/SQL Packages and Types Reference

Automatic Refresh of Cached Message with a Cache Function

The DBMS_PIPE package allows you to automatically populate a Singleton Pipe message using a user-defined cache function.

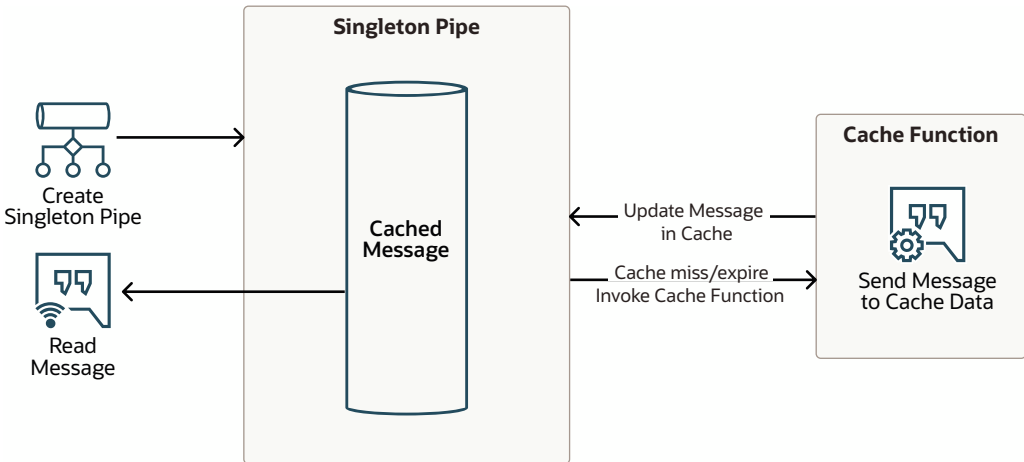
By default, after a message is invalidated with either Singleton Pipe explicit or implicit invalidation, a subsequent DBMS_PIPE.RECEIVE_MESSAGE results in no message being received. To add a new message to the pipe, the message must be explicitly cached by calling DBMS_PIPE.SEND_MESSAGE. To avoid this case, where no message is available when you read from a Singleton Pipe, you can define a cache function. With a cache function defined, the cache function is automatically invoked when you receive a message in following scenarios:

- When the Singleton Pipe is empty.
- When the message in a Singleton Pipe is invalid due to the shelfLife time elapsed.

To use a cache function define the cache function and include the cache_func parameter with DBMS_PIPE.RECEIVE_MESSAGE. A user-defined cache function provides the following:

- The cache function can be specified when reading a message from a Singleton Pipe using DBMS_PIPE.RECEIVE_MESSAGE.
- When there is no message in the Singleton Pipe, DBMS_PIPE.RECEIVE_MESSAGE calls the cache function.
- When the message shelfLife time has elapsed, the database automatically populates a new message in the Singleton Pipe.

Using a cache function simplifies working with Singleton Pipes. You do not need to handle failure cases for receiving a message from an empty pipe. In addition, a cache function ensures there is no cache-miss when you read messages from a Singleton Pipe, providing maximum use of the cached message.



When you define a cache function, the function name must be fully qualified with the owner schema:

- `OWNER.FUNCTION_NAME`

- `OWNER.PACKAGE.FUNCTION_NAME`

Define a cache function with the following signature:

```
CREATE OR REPLACE FUNCTION cache_function_name(
    pipename IN VARCHAR2
) RETURN INTEGER;
```

The typical operations within a cache function are:

- Create a Singleton Pipe, for an Explicit Pipe, using `DBMS_PIPE.CREATE_PIPE`.
- Create a message to cache in the Singleton Pipe.
- Send the message to the pipe specified in the cache function, optionally specifying a `shelflife` for the implicit message.

To use a cache function, the current session user that invokes `DBMS_PIPE.RECEIVE_MESSAGE` must have required privileges to execute the cache function.

See `RECIEVE_MESSAGE` Function more information on defining a cache function.

Create an Explicit Singleton Pipe

Describes the steps to create a Singleton Pipe with a specified pipe name (an Explicit Singleton Pipe).

First, for this example create the `receive_message` helper function to repeatedly call `DBMS_PIPE.RECEIVE_MESSAGE`. This allows you to test singleton pipe functionality.

```
CREATE OR REPLACE FUNCTION msg_types AS
    TYPE t_rcv_row IS RECORD (c1 VARCHAR2(32767), c2 NUMBER);
    TYPE t_rcv_tab IS TABLE OF t_rcv_row;
END;

CREATE OR REPLACE FUNCTION receive_message(
    pipename    IN VARCHAR2,
    rcv_count   IN NUMBER DEFAULT 1,
    cache_func  IN VARCHAR2 DEFAULT NULL)
RETURN msg_types.t_rcv_tab pipelined
AS
    l_msg    VARCHAR2(32767);
    l_status NUMBER;
BEGIN
    FOR i IN 1..rcv_count LOOP
        l_status := DBMS_PIPE.RECEIVE_MESSAGE(
            pipename => pipename,
            cache_func => cache_func,
            timeout   => 1);
        IF l_status != 0 THEN
            raise_application_error(-20000,
                'Message not received for attempt: ' || to_char(i) || ' status: ' ||
                l_status);
        END IF;

        DBMS_PIPE.UNPACK_MESSAGE(l_msg);
```

```

        pipe row(msg_types.t_rcv_row(l_msg));
    END LOOP;
RETURN;
END;

```

1. Create an explicit singleton pipe named PIPE_TEST with shelflife parameter set to 3600 (seconds).

```

DECLARE
    l_status INTEGER;
BEGIN
    l_status := DBMS_PIPE.CREATE_PIPE(
        pipename => 'MY_PIPE1',
        private => TRUE,
        singleton => TRUE,
        shelflife => 3600);
END;
/

```

See CREATE_PIPE Function for more information.

2. Verify the singleton pipe is created.

```

SELECT name, singleton, type
       FROM v$db_pipes WHERE name= '&pipename' ORDER BY 1;

```

NAME	SINGLETON	TYPE
-----	-----	-----
PIPE_TEST	YES	PRIVATE

3. Pack and send a message on the singleton pipe.

```

EXEC DBMS_PIPE.PACK_MESSAGE('This is a real message that you can get
multiple times');

```

```

SELECT DBMS_PIPE.SEND_MESSAGE(pipename => '&pipename') status FROM DUAL;

```

```

STATUS
-----
0

```

See *PACK_MESSAGE Procedures* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) and SEND_MESSAGE Function for more information.

4. Receive a message from a singleton pipe.

```

SELECT * FROM receive_message(
    pipename => '&pipename',
    rcv_count => 2);

```

```

MESSAGE
-----
-----

```

This is a real message that you can get multiple times
This is a real message that you can get multiple times

The `receive_message` function is a helper function that calls `DBMS_PIPE.RECEIVE_MESSAGE`.

5. Purge the message and remove the pipe.

```
EXEC DBMS_PIPE.PURGE('&pipename');
SELECT DBMS_PIPE.REMOVE_PIPE('&pipename') status FROM DUAL;
```

Create an Explicit Singleton Pipe with a Cache Function

Describes the steps to create a Singleton Pipe with a specified pipe name, an Explicit Singleton Pipe, and provide a cache function. A cache function allows you to automatically populate the message in a singleton pipe.

1. Create a cache function, `test_cache_message` for a singleton pipe.

```
CREATE OR REPLACE FUNCTION test_cache_message(
    pipename IN VARCHAR2) return NUMBER
AS
    l_status NUMBER;
    l_data VARCHAR2(4000);
BEGIN
    l_status := DBMS_PIPE.CREATE_PIPE(
        pipename => pipename,
        private => TRUE,
        singleton => true,
        shelflife => 600);
    IF l_status != 0 THEN RETURN l_status;
    END IF;

    DBMS_PIPE.PACK_MESSAGE('This is a placeholder cache message for an
empty pipe');
    l_status := DBMS_PIPE.SEND_MESSAGE(pipename => pipename);
    RETURN l_status;
END;
/
```

Note

The current session user invoking `DBMS_PIPE.RECEIVE_MESSAGE` must have required privilege to execute the cache function.

2. Receive with a cache function and confirm the message populates in pipe. The pipe must exist as a private pipe created in the cache function.

```
SELECT * FROM receive_message(
    pipename => '&pipename',
    rcv_count => 1,
    cache_func => 'TEST_CACHE_MESSAGE');

MESSAGE
```

This is a placeholder cache message for an empty pipe

The `receive_message` function is a helper function that calls `DBMS_PIPE.RECEIVE_MESSAGE`. See [Create an Explicit Singleton Pipe](#) for the `receive_message` definition.

See `CREATE_PIPE` Function for more information.

3. Receive without the cache function to confirm the message persists in the pipe.

```
SELECT * FROM receive_message(
    pipename => '&pipename',
    rcv_count => 2);
```

MESSAGE

This is a placeholder cache message for an empty pipe

This is a placeholder cache message for an empty pipe

The `receive_message` function is a helper function that calls `DBMS_PIPE.RECEIVE_MESSAGE`. See [Create an Explicit Singleton Pipe](#) for the `receive_message` definition.

See `CREATE_PIPE` Function for more information.

Use Persistent Messaging with Messages Stored in Cloud Object Store

The `DBMS_PIPE` package has extended functionality on Autonomous AI Database to support persistent messaging, where messages are stored in Cloud Object Store.

About Persistent Messaging with `DBMS_PIPE`

Persistent messaging with `DBMS_PIPE` allows one or more database sessions to communicate in the same region or across regions with messages that are stored in Cloud Object Store. Persistent messages in `DBMS_PIPE`:

- Allow you to send and retrieve very large messages.
- Support a sending a large number of pipe messages.
- Support sending and receiving messages within a single database, across multiple databases and across databases in different regions.
- Support multiple pipes using the same Cloud Object Store location URI.

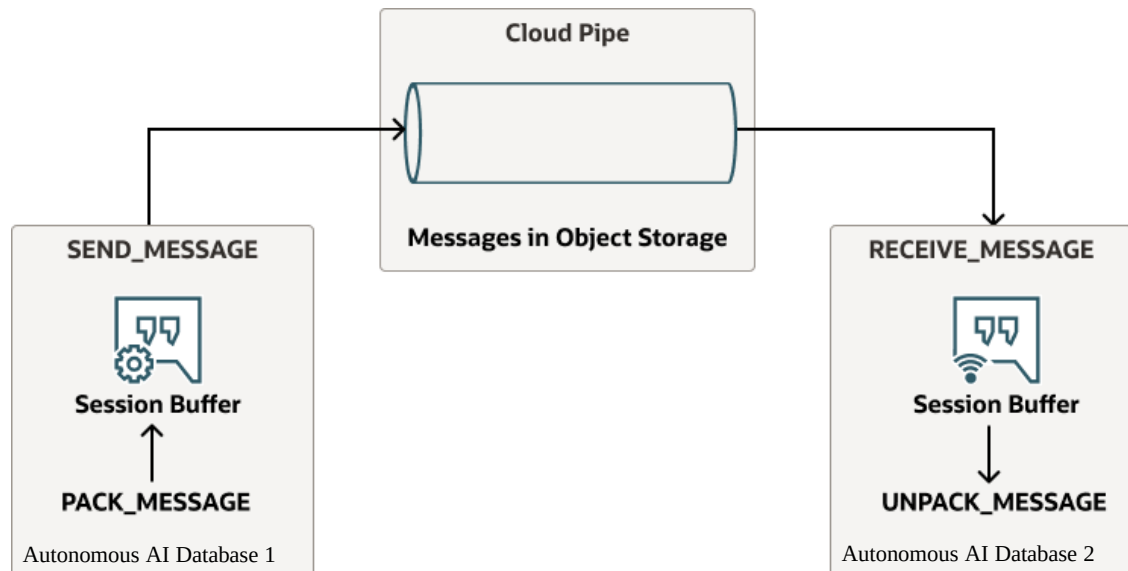
Persistent messaging pipes can be created in any of the supported `DBMS_PIPE` types:

- **Implicit Pipe:** Automatically created when a message is sent with an unknown pipe name using the `DBMS_PIPE.SEND_MESSAGE` function.
- **Explicit Pipe:** Created using the `DBMS_PIPE.CREATE_PIPE` function with a user specified pipe name.
- **Public Pipe:** Accessible by any user with `EXECUTE` permission on `DBMS_PIPE` package.
- **Private Pipe:** Accessible by sessions with the same user as the pipe creator.

Note

Oracle recommends creating an explicit pipe before you send or receive messages with persistent messaging. Creating an explicit pipe with `DBMS_PIPE.CREATE_PIPE` ensures that the pipe is created with the access permissions you want, either public or private (by setting the `private` parameter).

The following shows the general workflow for `DBMS_PIPE` with persistent messaging:



Existing applications using `DBMS_PIPE` can continue to operate with minimal changes. You can configure existing applications that use `DBMS_PIPE` with a credential object and location URI using a logon trigger or using some other initialization routine. After setting the `DBMS_PIPE` credential and location URI, no other changes are needed to use persistent messaging. All subsequent use of the pipe stores the messages in Cloud Object Store instead of in database memory. This allows you to change the storage method for messages from in-memory to persistent Cloud Object Storage, with minimal changes.

Persistent Messaging Overview and Features

Features of `DBMS_PIPE` using persistent messaging:

- Messages can be sent and retrieved across multiple Autonomous AI Database instances in the same region or across regions.
- Persistent messages are guaranteed to either be written or read by exactly one process. This prevents message content inconsistency due to concurrent reads and writes. `DBMS_PIPE` allows only one operation, sending a message or receiving message to be active at a given time and these operations are protected by a locking mechanism. However, if an operation is not possible due to an ongoing operation, the process retries periodically until the timeout value is reached.
- `DBMS_PIPE` uses `DBMS_CLOUD` to access Cloud Object Store. Messages can be stored in any of the supported Cloud Object Stores. See Cloud Object Storage URI Formats for more information.
- `DBMS_PIPE` uses `DBMS_CLOUD` to access Cloud Object Store and all supported credential types are available:

- `DBMS_CLOUD.CREATE_CREDENTIAL`: See `CREATE_CREDENTIAL` procedure for more information.

DBMS_PIPE Privileges Authorization and Security

The `DBMS_PIPE` procedures run with invoker's rights. Private pipes are owned by the current user and a private pipe that is created by a user can only be used by the same user. This applies to both in-memory pipes and persistent messaging pipes where messages are stored to Cloud Object Store. Sending and receiving messages run in the invoker's schema.

Using private pipes, where messages are stored to Cloud Object Store, a credential object is required for authentication with the Cloud Object store identified by the `location_uri` parameter. The invoking user must have `EXECUTE` privilege on the credential object specified with the `credential_name` parameter that is used to access the Object Store.

To use a public pipe, the user, database session, must have `execute` privilege on `DBMS_PIPE`. For a public pipe using persistent messaging and storing messages to Cloud Object Store, the user, database session, must have `execute` privilege on `DBMS_CLOUD` and `execute` privilege on the credential object (or you can create a credential object that is allowed to access the location URI that contains the message).

DBMS_PIPE Limitation

The `DBMS_PIPE` package does not support sending messages between databases that use different character sets. For example, if you have one Autonomous AI Database instances that uses `AL32UTF8` and another instance that uses `WE8MSWIN1252`, you cannot send messages with `DBMS_PIPE` between these two databases. In this case, you the system will raise the error `ORA-12704` if you attempt to send messages with `DBMS_PIPE` between these two databases.

See Character Set Selection for Autonomous AI Database for more information.

Create an Explicit Persistent Pipe and Send a Message

Describes the steps to create a persistent pipe with a specified pipe name (Explicit Pipe).

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

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

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. After you store the credentials you can then use the same credential name to access Cloud Object Store to send and receive messages with `DBMS_PIPE`.

For detailed information about the parameters, see `CREATE_CREDENTIAL` Procedure. For Oracle Cloud Infrastructure Object Storage, it is required that the credential uses native Oracle Cloud Infrastructure authentication.

Note

Some tools like SQL*Plus and SQL Developer use the ampersand character (&) as a special character. If you have the ampersand character in your password use the SET DEFINE OFF command in those tools as shown in the example to disable the special character and get the credential created properly.

2. Create an explicit pipe to send and retrieve messages. For example, create a pipe named ORDER_PIPE.

```
DECLARE
  r_status INTEGER;
BEGIN
  r_status := DBMS_PIPE.CREATE_PIPE(pipename => 'ORDER_PIPE');
END;
/
```

See CREATE_PIPE Function for more information.

3. Verify that the pipe is created.

```
SELECT ownerid, name, type FROM v$db_pipes
       WHERE name = 'ORDER_PIPE';
```

```
OWNERID NAME      TYPE
-----
      80 ORDER_PIPE PRIVATE
```

4. Use DBMS_PIPE procedures to set the default access credential and location URI to store persistent messages to Cloud Object Store.

```
BEGIN
  DBMS_PIPE.SET_CREDENTIAL_NAME('my_persistent_pipe_cred');
  DBMS_PIPE.SET_LOCATION_URI('https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespaces-string/b/bucketname1/');
END;
/
```

These procedures set the default credential name and the default location URI for use with DBMS_PIPE procedures.

If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:

- If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

See SET_CREDENTIAL_NAME Procedure and GET_LOCATION_URI Function for more information.

5. Pack and send a message on the pipe.

```

DECLARE
    l_result INTEGER;
    l_date   DATE;
BEGIN
    l_date := sysdate;
    DBMS_PIPE.PACK_MESSAGE(l_date);           -- date of order
    DBMS_PIPE.PACK_MESSAGE('C123');          -- order number
    DBMS_PIPE.PACK_MESSAGE(5);               -- number of items in order
    DBMS_PIPE.PACK_MESSAGE('Printers');      -- type of item in order

    l_result := DBMS_PIPE.SEND_MESSAGE(
        pipename => 'ORDER_PIPE',
        credential_name => DBMS_PIPE.GET_CREDENTIAL_NAME,
        location_uri => DBMS_PIPE.GET_LOCATION_URI);

    IF l_result = 0 THEN
        DBMS_OUTPUT.put_line('DBMS_PIPE sent order successfully');
    END IF;

END;
/

```

See *PACK_MESSAGE Procedures* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) and *SEND_MESSAGE Function* for more information.

Retrieve a Persistent Message on Same Database

Describes the steps to retrieve a persistent message from an explicit pipe on the same Autonomous AI Database instance (the instance where the message was sent).

On an Autonomous AI Database instance you can receive messages sent to a pipe from a different session. The DBMS_PIPE procedures are invoker's rights procedures and run as the current invoked user.

Private pipes are owned by the current user that creates the pipe. Private pipes can only be accessed by the same user that created the pipe. This applies to pipes using in-memory messages and to pipes using persistent messaging with messages stored in Cloud Object Store.

Public pipes can be accessed by any database session having execute privilege on DBMS_PIPE. This applies to pipes using in-memory messages and to pipes using persistent messaging with messages stored in Cloud Object Store.

1. Verify that the pipe is created.

```

SELECT ownerid, name, type FROM v$db_pipes
       WHERE name = 'ORDER_PIPE';

```

```

OWNERID NAME          TYPE
-----
80 ORDER_PIPE PRIVATE

```

When you are on the same Autonomous AI Database instance and the pipe exists, you do not need to run `DBMS_PIPE.CREATE_PIPE` before you receive a message. This applies when the pipe was created on the same instance, as shown in [Create an Explicit Persistent Pipe and Send a Message](#).

2. Receive a message from the pipe.

```
DECLARE
    message1  DATE;
    message2  VARCHAR2(100);
    message3  INTEGER;
    message4  VARCHAR2(100);
    l_result  INTEGER;

BEGIN

    DBMS_PIPE.SET_CREDENTIAL_NAME('my_persistent_pipe_cred');
    DBMS_PIPE.SET_LOCATION_URI('https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/');
    l_result := DBMS_PIPE.RECEIVE_MESSAGE (
        pipename => 'ORDER_PIPE',
        timeout  => DBMS_PIPE.MAXWAIT,
        credential_name => DBMS_PIPE.GET_CREDENTIAL_NAME,
        location_uri => DBMS_PIPE.GET_LOCATION_URI);

    IF l_result = 0 THEN
        DBMS_PIPE.unpack_message(message1);
        DBMS_PIPE.unpack_message(message2);
        DBMS_PIPE.unpack_message(message3);
        DBMS_PIPE.unpack_message(message4);

        DBMS_OUTPUT.put_line('Order Received Successfully On: ' ||
TO_CHAR(sysdate, 'dd-mm-yyyy hh24:mi:ss'));
        DBMS_OUTPUT.put_line('Date of Order: ' || message1);
        DBMS_OUTPUT.put_line('Order Number: ' || message2);
        DBMS_OUTPUT.put_line('Number of Items In Order: ' || message3);
        DBMS_OUTPUT.put_line('Item Type in Order: ' || message4);
    END IF;

END;
/
```

When you are on the same Autonomous AI Database instance, the credential already exists and you do not need to run `DBMS_CLOUD.CREATE_CREDENTIAL` to receive a message. This applies when the pipe was created on the same instance, as shown in [Create an Explicit Persistent Pipe and Send a Message](#).

See `SET_CREDENTIAL_NAME` Procedure and `GET_LOCATION_URI` Function for more information.

See `RECEIVE_MESSAGE` Function for more information.

Retrieve a Persistent Message by Creating a Pipe on a Different Database

Describes the steps to retrieve a persistent message stored in Cloud Object Store with an explicit pipe on an Autonomous AI Database instance that is different than the instance that sent the message.

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

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

This operation stores the credentials in the database in an encrypted format. You can use any name for the credential name. Note that this step is required only once unless your object store credentials change. Once you store the credentials you can then use the same credential name to access the Cloud Object Store to send and receive messages with `DBMS_PIPE`.

For detailed information about the parameters, see `CREATE_CREDENTIAL` Procedure.

Note

Some tools like SQL*Plus and SQL Developer use the ampersand character (&) as a special character. If you have the ampersand character in your password use the `SET DEFINE OFF` command in those tools as shown in the example to disable the special character and get the credential created properly.

2. Create an explicit pipe with the same name as the pipe that sent the message. For example, create a pipe named `ORDER_PIPE`.

```
DECLARE
  r_status INTEGER;
BEGIN
  r_status := DBMS_PIPE.CREATE_PIPE(pipename => 'ORDER_PIPE');
END;
/
```

See `CREATE_PIPE` Function.

3. Verify that the pipe is created.

```
SELECT ownerid, name, type FROM v$db_pipes
       WHERE name = 'ORDER_PIPE';
```

OWNERID	NAME	TYPE

```
-----
      80 ORDER_PIPE PRIVATE
```

4. Use DBMS_PIPE procedures to set the default access credential and location URI for Object Store so that DBMS_PIPE can access the persistent message.

```
BEGIN
    DBMS_PIPE.SET_CREDENTIAL_NAME('my_persistent_pipe_cred');
    DBMS_PIPE.SET_LOCATION_URI('https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/');
END;
/
```

These procedures set the default credential name and the default location URI for use with DBMS_PIPE procedures.

If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:

- If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

See SET_CREDENTIAL_NAME Procedure and GET_LOCATION_URI Function for more information.

5. Receive a message from the persistent pipe.

```
DECLARE
    message1  DATE;
    message2  VARCHAR2(100);
    message3  INTEGER;
    message4  VARCHAR2(100);
    l_result  INTEGER;

BEGIN

    DBMS_PIPE.SET_CREDENTIAL_NAME('my_persistent_pipe_cred');
    DBMS_PIPE.SET_LOCATION_URI('https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/');
    l_result := DBMS_PIPE.RECEIVE_MESSAGE (
        pipename => 'ORDER_PIPE',
        timeout  => DBMS_PIPE.MAXWAIT,
        credential_name => DBMS_PIPE.GET_CREDENTIAL_NAME,
        location_uri => DBMS_PIPE.GET_LOCATION_URI);

    IF l_result = 0 THEN
        DBMS_PIPE.unpack_message(message1);
        DBMS_PIPE.unpack_message(message2);
        DBMS_PIPE.unpack_message(message3);
        DBMS_PIPE.unpack_message(message4);

        DBMS_OUTPUT.put_line('Order Received Successfully On: ' ||
TO_CHAR(sysdate, 'dd-mm-yyyy hh24:mi:ss'));
```

```
        DBMS_OUTPUT.put_line('Date of Order: ' || message1);
        DBMS_OUTPUT.put_line('Order Number: ' || message2);
        DBMS_OUTPUT.put_line('Number of Items In Order: ' || message3);
        DBMS_OUTPUT.put_line('Item Type in Order: ' || message4);
    END IF;

END;
/
```

See `RECEIVE_MESSAGE` Function for more information.

Remove a Persistent Pipe

Describes the steps to remove a persistent pipe.

Persistent pipes send and receive messages by storing messages in Cloud Object Store. Use `DBMS_PIPE.REMOVE_PIPE` to remove a persistent pipe on an Autonomous AI Database instance.

1. Call the `DBMS_PIPE.REMOVE_PIPE` function to remove a pipe.

```
DECLARE
    l_result INTEGER;
BEGIN
    l_result := DBMS_PIPE.REMOVE_PIPE('ORDER_PIPE');
END;
/
```

The `REMOVE_PIPE` function removes the pipe from the Autonomous AI Database instance where it runs, however `REMOVE_PIPE` does not affect other Autonomous AI Database instances with a pipe with the same name that uses the same location URI.

2. On the Autonomous AI Database instance where you run `DBMS_PIPE.REMOVE_PIPE`, verify that the pipe is removed.

```
SELECT ownerid, name, type FROM v$db_pipes
       WHERE name = 'ORDER_PIPE';
```

No rows selected

Invoke Cloud Functions and External Procedures

In an on-premise Oracle Database (Enterprise, Standard, Express, Personal Editions), you can write user-defined functions in PL/SQL, Java, or C to provide functionality that is not available in SQL or SQL built-in functions. You can invoke these user-defined functions in a SQL statement wherever an expression can occur. See [About User Defined Functions](#).

From Oracle Database 19c onwards, you can create, store, and execute Python scripts using the Embedded Python execution feature in SQL. See [Embedded Python Execution using OML4Py](#).

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure extends user defined functions capability to the cloud by enabling you to write SQL Functions that invoke serverless compute services in the cloud, such as OCI Functions and AWS Lambda Functions, or

External Procedures - which include C/C++ routines, Shell or Python scripts - executing in an OCI compute VM that is external to the Autonomous AI Database.

About Cloud Functions and External Procedures

User defined functions enable you to invoke externally available functions from PL/SQL or SQL code within your database. You can invoke the following external functions using user defined functions:

- **Oracle Cloud Infrastructure Functions:** Oracle Cloud Infrastructure Functions are fully managed, multi-tenant, highly scalable, on-demand, Functions-as-a-Service platform. Oracle Cloud Infrastructure Functions are built on enterprise-grade Oracle Cloud Infrastructure and powered by the Fn Project open-source engine. See [Overview of OCI Functions](#) for more information.
- **AWS Lambda Functions:** AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any application or backend service without provisioning or managing servers. See [AWS Lambda](#) for more information.
- **External Procedures:** External procedures are functions written in a third-generation language (C, for example), or OS shell scripts or Python scripts that can be invoked as SQL functions or PL/SQL procedures or functions. See [What is an External Procedure](#) for more information.

Invoke OCI Cloud Functions as SQL Functions

Shows the steps to invoke OCI cloud functions as SQL functions in your Autonomous AI Database on Dedicated Exadata Infrastructure.

Before you proceed with these steps, it is assumed that you have created and deployed OCI Functions in a OCI tenancy and compartment. See [OCI Functions](#) for details.

Once you have working OCI Function(s), you will be using the DBMS_CLOUD and DBMS_CLOUD_FUNCTION PL/SQL APIs to create a catalog of SQL wrapper functions in the Autonomous AI Database that reference and call their respective cloud function via their API endpoints. You will be using DBMS_CLOUD_FUNCTION API to manage the functions from your database application.

1. Create credentials using the DBMS_CLOUD.CREATE_CREDENTIAL Procedure.

Provide the OCI tenancy user's API Private key (content of the dot-pem file that you downloaded when creating the API key) as the credential object, the key's fingerprint, the tenancy OCID, and the user OCID. See [Required Keys and OCIDs](#).

```
SET DEFINE OFF
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL (
        credential_name => 'OCI_CRED', -- provide a string as the name
        user_ocid       => 'user_ocid', -- provide the OCID string for the
user, obtained from OCI Console User Profile
        tenancy_ocid    => 'tenancy_ocid', -- provide the OCID string for
the tenancy, obtained from OCI Console User Profile
        private_key     => 'private_key', -- provide the content of the dot-
pem file that you downloaded when you created the API Key
        fingerprint     => 'fingerprint' -- provide the fingerprint string
for the API key
    );
```

```
END;
/
```

This creates the OCI_CRED credential object.

See CREATE_CREDENTIAL Procedure for more information.

2. Create a Catalog object using the DBMS_CLOUD_FUNCTION.CREATE_CATALOG Procedure.

A catalog is a collection of wrapper functions that reference and call their respective cloud functions via their API endpoints. Provide the credential object, the cloud service provider name - in this case OCI, and the OCI Region ID (PHX in this example), and OCI Compartment ID in which the OCI Functions are located.

```
BEGIN
DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
    credential_name => 'OCI_CRED',
    catalog_name    => 'OCI_DEMO_CATALOG',
    service_provider => 'OCI',
    cloud_params    => '{"region_id":"phx",
"compartment_id":"compartment_id"}'
);
END;
/
```

This creates the OCI_DEMO_CATALOG catalog object.

See CREATE_CATALOG Procedure for more information. You can query DBA_CLOUD_FUNCTION_CATALOG View and USER_CLOUD_FUNCTION_CATALOG View to retrieve the list of all the cloud function catalogs in your database.

3. You can list all the cloud functions in the catalog using the DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS Procedure.

```
SET PAGESIZE 1000
```

```
VAR function_list CLOB;
```

```
BEGIN
    DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS (
        credential_name => 'OCI_CRED',
        catalog_name    => 'OCI_DEMO_CATALOG',
        function_list    => :function_list
    );
END;
/
```

PL/SQL procedure successfully completed.

```
SELECT JSON_QUERY (:function_list, '$' RETURNING VARCHAR2(32676) pretty)
AS search_results FROM dual;
```

```
SEARCH_RESULTS
```

```
-----
-----
[
```



```

{
  "functionName" : "create_par",
  "functionId" : "ocid.funfc.oc1.phx.aaaa_example",
  "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud_example.com"
},
{
  "functionName" : "fintech",
  "functionId" : "ocid.funfc.oc1.phx.bbbb_example",
  "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud.com_example"
},
{
  "functionName" : "jwt_codec",
  "functionId" : "ocid.funfc.oc1.phx.jwt_code_example",
  "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud_example.com"
},
{
  "functionName" : "oci-objectstorage-create-par-python",
  "functionId" : "ocid.funfc.oc1.phx.aaaaaaaas_example",
  "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud_example.com"
},
{
  "functionName" : "run_dbt",
  "functionId" : "ocid.funfc.oc1.phx.aaaaaaaav_example",
  "invokeEndpoint" : "https://dw.us.func.oci.oraclecloud_example.com"
}
]

```

See LIST_FUNCTIONS Procedure for more information.

4. Create the SQL Wrapper Function. You can use one of the following two methods to create the wrapper SQL functions from the catalog, that call their respective cloud functions:
 - You can use DBMS_CLOUD_FUNCTION. SYNC_FUNCTIONS Procedure to generate SQL wrappers from the catalog object.
 - You can manually create individual wrapper functions using DBMS_CLOUD_FUNCTION. CREATE_FUNCTION Procedure.
- a. **SYNC_FUNCTIONS:** SYNC_FUNCTIONS automatically syncs (creates and/or updates) wrapper functions in the catalog with the complete list of cloud functions defined in the region, compartment, and tenancy with which the catalog was created.

```

BEGIN
  DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS (
    catalog_name => 'OCI_DEMO_CATALOG'
  );
END;
/

```

This creates a PL/SQL wrapper for adding new functions to the catalog and removing wrappers for functions that have been deleted from the catalog.

You can verify the sync results using the following query, for a given current user.

```

SHOW user
TEST_USER

```

```
SELECT object_name FROM sys.all_objects WHERE owner='TEST_USER' AND
object_type='FUNCTION';
```

```
OBJECT_NAME
```

```
-----
```

```
CREATE_PAR
FINTECH
JWT_CODEC
OCI-OBJECTSTORAGE-CREATE-PAR-PYTHON
RUN_DBT
```

See SYNC_FUNCTIONS Procedure for more information.

- b. **CREATE_FUNCTION:** You can manually create a SQL Function in your catalog that calls its respective cloud function using DBMS_CLOUD.CREATE_FUNCTION.

```
VAR function_args CLOB;
```

```
EXEC :function_args := TO_CLOB('{ "command": "VARCHAR2", "value":
"VARCHAR2"}');
```

```
BEGIN
```

```
  DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    credential_name => 'OCI_CRED',
    catalog_name    => 'OCI_DEMO_CATALOG',
    function_name    => 'FINTECH_FUNCTION',
    function_id      => 'ocid1.fnfunc.oc1.phx.aaabbbccccc_example',
    input_args       => :function_args
```

```
  );
END;
/
```

This creates the FINTECH_FUN function in the OCI_DEMO_CATALOG catalog, as a reference to the respective cloud function whose endpoint is referenced by the FUNCTION_ID parameter. Invoking the function in the catalog along with its arguments runs the corresponding cloud function in OCI, and provides the output returned by the cloud function.

Note

The name of the OCI Function can be something entirely different than FINTECH_FUNCTION. Only the OCID of the OCI Function that you provide as input to the function_id parameter is taken into account to create this reference.

- c. **CREATE_FUNCTION with custom return types and response handlers:** Manual creation of functions allows you to create custom return types and response handlers, and shown in the following example.

First create a return type and the function's response handler.

```
CREATE OR REPLACE TYPE fintech_rt AS OBJECT (
  status VARCHAR2(1000),
  output CLOB
```

```

);
/
Type created.

CREATE OR REPLACE FUNCTION fintech_response_handler(function_response
IN CLOB)
RETURN fintech_rt
IS
    l_comp      fintech_rt;
    l_json_obj  JSON_OBJECT_T;
    status      VARCHAR2(1000);
    output      CLOB;
BEGIN
    l_json_obj := JSON_OBJECT_T.parse(function_response);
    status     := l_json_obj.get('STATUS').to_string;
    output     := l_json_obj.get('RESPONSE_BODY').to_string;
    l_comp     := fintech_rt(status,output);
    RETURN l_comp;
END;
/
Function created.

```

Next, use this type and response handler during the manual creation of the SQL Wrapper Function.

```

VAR input_param CLOB;
VAR l_return_type VARCHAR2(100);
VAR l_response_handler VARCHAR2(1000);

-- Define function parameters
exec :input_param      := TO_CLOB('{"command": "VARCHAR2", "value":
"VARCHAR2"}');

PL/SQL procedure successfully completed.

exec :l_return_type     := 'fintech_rt';

PL/SQL procedure successfully completed.

exec :l_response_handler := 'fintech_response_handler';

PL/SQL procedure successfully completed.

BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
        credential_name => 'OCI_CRED',
        catalog_name    => 'OCI_DEMO_CATALOG',
        function_name    => 'FINTECH_FUNCTION',
        function_id      => 'ocid1.fnfunc.oc1.phx.aaabbbccccc_example',
        input_args       => :input_param,
        return_type      => :l_return_type,
        response_handler => :l_response_handler
    );
END;
/

```

You can query `DBA_CLOUD_FUNCTION` View and `USER_CLOUD_FUNCTION` View views to retrieve the list of all the functions in your database.

See `CREATE_FUNCTION` Procedure for more information.

Once the function is created, you can `DESCRIBE` the function to get its return details.

```
DESC fintech_fun
COLUMN STATUS format a30
COLUMN OUTPUT format a30
```

You can then invoke the function, providing the values for input parameters.

```
SET SERVEROUTPUT ON

DECLARE
    l_comp fintech_rt;
BEGIN
    l_comp := fintech_fun(command=>'tokenize',value =>
'PHI_INFORMATION');

    DBMS_OUTPUT.put_line ('Status of the function   =  '||
l_comp.status);
    DBMS_OUTPUT.put_line ('Response of the function =  '||
l_comp.output);
END;
/
```

This invokes the `fintech_fun` cloud function by calling the function reference `oocid1.funfn.oci.phx.aaaaaa_example` in the `OCI_DEMO_CATALOG` catalog.

5. You can drop an existing function using the `DROP_FUNCTION` procedure.

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_FUNCTION (
        catalog_name => 'OCI_DEMO_CATALOG',
        function_name => 'fintech_fun');
END;
/
```

This drops the `FINTECH_FUN` function from the `OCI_DEMO_CATALOG` catalog.

See `DROP_FUNCTION` Procedure for more information.

6. You can drop an existing catalog using the `DROP_CATALOG` procedure.

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_CATALOG (
        catalog_name => 'OCI_DEMO_CATALOG'
    );
END;
/
```

This drops the `OCI_DEMO_CATALOG` from your database.

See `DROP_CATALOG` Procedure for more information.

Invoke AWS Lambda Functions as SQL Functions

Shows the steps to invoke AWS remote functions as SQL functions in your Autonomous AI Database on Dedicated Exadata Infrastructure.

Before you proceed with these steps, it is assumed that you have created and deployed AWS Lambda Functions in a AWS tenancy. See [AWS Lambda](#) for details.

To access AWS lambda functions you need to configure the necessary policies in Oracle Cloud Infrastructure. See [Creating an IAM policy to access AWS Lambda resources](#) and [Using resource-based policies for Lambda](#) for more information.

Once you have working AWS Lambda function(s), you will be using the DBMS_CLOUD and DBMS_CLOUD_FUNCTION PL/SQL APIs to create a catalog of SQL wrapper functions in the Autonomous AI Database that reference and call their respective cloud function via their API endpoints. You will be using DBMS_CLOUD_FUNCTION API to manage the functions from your database application.

1. Create a credential using the procedure DBMS_CLOUD.CREATE_CREDENTIAL using the AWS Secret Key as the credential object.

```
SET DEFINE OFF
```

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'AWS_CRED',
    username       => 'access_key_ID', -- ID of Secret Key
    password       => 'secret_access_key' -- Secret Key password
  );
END;
/
```

This creates the AWS_CRED credential object.

See CREATE_CREDENTIAL Procedure for more information.

2. Create a Catalog object using the DBMS_CLOUD_FUNCTION.CREATE_CATALOG Procedure.

A catalog is a collection of wrapper functions that reference and call their respective cloud functions via their API endpoints. Provide the credential object, the cloud service provider name - in this case AWS, and the AWS Region ID (ap-northeast-1 in this example) in which the AWS Lambda Functions are located.

```
BEGIN
  DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
    credential_name => 'AWS_CRED',
    catalog_name    => 'AWS_DEMO_CATALOG',
    service_provider => 'AWS',
    cloud_params    => '{"region_id":"ap-northeast-1"}'
  );
END;
/
```

This creates the AWS_DEMO_CATALOG catalog object.

See `CREATE_CATALOG` Procedure for more information. You can query `DBA_CLOUD_FUNCTION_CATALOG` View and `USER_CLOUD_FUNCTION_CATALOG` View to retrieve the list of all the catalogs in your database.

The rest of the steps/operations - namely, `LIST_FUNCTIONS` Procedure, `SYNC_FUNCTIONS` Procedure, `CREATE_FUNCTION` Procedure, `DROP_FUNCTION` Procedure, and `DROP_CATALOG` Procedure - have the same usage as described for OCI cloud functions. See [Invoke OCI Cloud Functions as SQL Functions](#).

External Procedures Overview

Provides an overview of External Procedures and their use in Autonomous AI Database on Dedicated Infrastructure applications.

External procedures are functions written in a third-generation language and invoked as SQL functions or PL/SQL procedures or functions. SQL and PL/SQL are best suited for fast and efficient data and transaction processing in the database. External procedures can complement SQL and PL/SQL by executing compute and memory-intensive tasks in a dedicated, external VM, and providing the results back to the database. Typical examples of such tasks are scientific and engineering problems whose computational libraries exist in the Linux OS (and are not easily portable to a data platform), offline data analysis, control of real-time devices and processes, and so on.

You can invoke and use external procedures in your Autonomous AI Database with user defined functions. Rather than executing these procedures within the database, you will place the executable code in a dedicated, custom OCI Linux VM that is provisioned with a container and runtime to execute C programs, Shell scripts, and Python scripts, along with Oracle SQL*Net libraries to enable remote SQL execution from the database.

Note

The remote host for external procedures must be an OCI Linux EXTPROC VM provisioned from the OCI Marketplace and configured in a suitable OCI Virtual Cloud Network (VCN). See [System Configuration in OCI for External Procedures and Scripts](#) for details.

Remote execution on OCI Linux EXTPROC VM is supported only by Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Public Cloud starting with version 19.30, and in Oracle Database 26ai, starting with version 23.26.1. This feature is not yet supported by Autonomous AI Database on Dedicated Exadata Infrastructure database on Exadata Cloud@Customer (ExaCC).

System Configuration for External Procedures and Scripts

Shows the steps to provision the EXTPROC VM, and the steps to configure OCI network and compute for executing external procedures.

Minimally, the architecture consists of the following resources:

- OCI Virtual Cloud Network provisioned in an OCI Compartment in the OCI tenancy.
- An Autonomous AI Database, created as an instance under an Autonomous Container Database (ACD), that is provisioned from an Autonomous Exadata VM Cluster (AVMC), that is created on the Exadata Infrastructure on Oracle Public Cloud.
- An EXTPROC VM provisioned as part of an EXTPROC Stack from the OCI Marketplace.

As a best practice, the AVM cluster will be placed in a private subnet in a VCN. The Default route table, and the Security Lists of the VCN will be configured as per the Autonomous AI Database requirements in the OCI tenancy. The VCN may have a Default Route table and one Default Security List, where you can define all the routing and security rules. Optionally, the VCN may be configured with a NAT gateway if Autonomous AI Database applications and other resources in the private subnet need Egress access to the Internet.

Follow these minimal OCI Networking configuration steps when you introduce the ExtProc VM into this topology.

- You must be able to login to your ExtProc VM to build the shared library (dot-so) with the external procedures implemented in C/C++, inspect and configure the VM and container, and perform other such operations.
- The typical practice is to configure the EXTPROC VM to be in a public subnet in the VCN. In this case, you must create an Internet Gateway in the VCN (if not already configured by the administrator).

Note

The EXTPROC VM must NOT be in a subnet whose CIDR range is 10.x.x.x. If the IP address of the EXPROC VM is 10.x.x.x, it will not be able to communicate with the Autonomous AI Database.

- You must define an Ingress rule in the VCN's Security List to allow for SSH login access to the VM via Port 22. You must also define an Egress rule in VCN's Security List to allow ADBD to access the Destination Port 16000 to be specific, or all destination ports, in general. See [OCI Networking Access and Security](#).
- As part of provisioning the EXTPROC VM Stack, you will provide all the Client IP addresses of the AVMC. You will obtain these IPv4 Client IP addresses from the OCI Dashboard for the AVM Cluster.
- Subsequently, the provisioning tool will automatically create a network security group (NSG) - named nsg_acl. This NSG will have an Ingress Rule allowing the Autonomous Database Dedicated Client IP addresses as the only sources that can access the Port 16000 destination.

Advanced Topics

- If the subnets hosting the ExtProc VM and Autonomous AI Database are in the same VCN, you need not specify any special routing rules in the VCN's routing table. If the subnets hosting the ExtProc VM and ADBD are in different VCNs, then you must implement VCN peering. See [VCN Peering](#).
- If you choose to configure the ExtProc VM to be in a private subnet, you must be able to login to the VM via a bastion/jump-off server and enable Ingress only into the ExtProc VM, and secure access to the other resources in the private subnet, such as the Autonomous AI Database. See the [OCI Networking Guides](#) for details.

Implement User Defined Functions with External Procedures

1. Provision and Configure the EXTPROC VM from the OCI Marketplace

- a. Sign in to the OCI Console at <http://cloud.oracle.com>. See [Sign in to the Oracle Cloud Infrastructure Console](#) for more information.

- b. From the Oracle Cloud Infrastructure left navigation menu, drop down to **Marketplace** and then, under Marketplace, click **All Applications**.
- c. Enter "EXTPROC" in the Search bar, and click Search. You will see two widgets - one labelled Stack, and another labelled Image.
- d. Choose the widget named **Stack** (this is important). This will take you to **Oracle Autonomous Database EXTPROC Agent details** page, providing the stack version, release date, and release notes about the Stack. On this page, click on the **Launch Stack** button. This takes you to the next Launch Stack page.
- e. On the Launch Stack page:
 - From the **Version** drop-down list, choose the package version of the stack.
 - From the **Compartment** drop-down list, select the name of the compartment where you want to provision the EXTPROC VM. Unless you have specific reasons not to, choose the same compartment as that of the ADBD (as a best practice, avoid the root compartment for any resource). Review [System Configuration for External Procedures and Scripts](#) to make the appropriate choice for the compartment.
 - Accept the terms and conditions, and click on **Launch Stack**.This takes you to the next page - the Create Stack wizard.
- f. In this page, provide a **name**, **description**, **Compartmentname**, the Terraform version, and any **Tag** information. The compartment name is mandatory, and most important, input to the wizard in this page. Click **Next**.
- g. This takes you to the configuration variables page for the EXTPROC VM (also called the EXTPROC Agent). The first part of this page gathers the information on the EXTPROC libraries and a wallet password.
 - For **External Libraries**, provide a list of libraries, separated by comma (,), that you want to allow to be invoked from your Autonomous AI Database on Dedicated Infrastructure. These shared libraries will contain the external procedures written in C/C++, that are ideally compiled and built in the VM itself (to match the architecture and environment).

Note

Given that you are still in the process of creating the EXTPROC VM, note down the names of the shared libraries that you provide here. When the VM is available, and you develop your external procedures in C/C++, you must remember to compile and build the library/libraries with the exact name(s) provided above.

- For **Wallet Password**, provide a password for the wallet file that will be created in the ExtProc VM. The wallet and a self-signed certificate is generated for mutual TLS authentication between the Autonomous AI Database on Dedicated Infrastructure and the EXTPROC Agent VM.
- h. The second part of the configuration variables page involves network configuration for the EXTPROC VM.
 - For **Compartment**, choose the compartment name for the VCN from the pulldown menu. In the above example, it is shown as adbd. Ideally, the Autonomous AI Database and the EXTPROC VM will be colocated in the same VCN, even if in different (private and public) subnets.

- For **Network Strategy**, choose **"Use Existing VCN and Subnet"**. As stated in a previous section, at this stage of your ExtProc development effort, you may have the system architecture in place with a VCN, AVMC, and Autonomous AI Database.

Note

The other choice is to create a new VCN and Subnet with its own configuration strategy for the EXTPROC VM Agent. As stated in the [System Configuration for External Procedures and Scripts](#), if you choose to place the EXTPROC VM in a VCN different than that of the Autonomous AI Database, you must peer the VCNs.

- Skip the next entry for **EXTPROC Agent Access Type**. This field is not applicable for Autonomous AI Database on Dedicated Exadata Infrastructure.
- Next, in the field labeled "Private Endpoint IP Addresses", enter ALL the Client IP addresses of the AVMC on which the database is configured. You can obtain these Client IP addresses from the OCI Console Dashboard for the AVMC.
 - For **Virtual Cloud Network**, provide the name of the VCN in which you want to create the EXTPROC VM Agent. Review [System Configuration for External Procedures and Scripts](#) to make the appropriate choice for the VCN.
 - For **EXTPROC Subnet**, provide the name of the subnet in which you want to create the EXTPROC VM Agent. Review [System Configuration for External Procedures and Scripts](#) to make the appropriate choice for the subnet.
 - Next, provide the details of the VCN and the Subnet.

Note

The EXTPROC VM must NOT be in a subnet whose CIDR range is 10.x.x.x. If the IP address of the EXPROC VM is 10.x.x.x, it will not be able to communicate with the Autonomous AI Database.

- The final part of the configuration page involves the compute configuration for the EXTPROC VM Agent.
 - For **Compartment**, choose the compartment name from the pulldown menu. It is recommended that you colocate the VM in the same compartment as that of the Autonomous AI Database on Dedicated Infrastructure, the VCN and the subnet to minimize troubleshooting.
 - For **Shape, Number of OCPUs**, and **Memory size (GBs)**, enter values based on the workload characteristics of your external procedures.
 - For **Add SSH keys**, generate the SSH public and private key files in your Linux OS, MacOS, or Windows system. Open a terminal window, and run one of the following commands (ED25519 is the modern, safer algorithm; older systems support the RSA encryption algorithm):

```
ssh-keygen -t ed25519 -C "your_email@example.com" (OR)
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

This will generate a private key file `ssh-key-<date-time-id>.key` and a public key file `ssh-key-<date-time-id>.key.pub`. Copy-paste the contents of the SSH public key into the window, or drag-drop the `.pub` file into the window.

- Click **Next**.

- I. This takes you to the Review Page. Review all the inputs, then select the **Run Apply** checkbox, and click on **Create**.

This will launch the OCI Resource Manager to create the EXTPROC VM Stack. Once the ORM Job is completed successfully, the EXTPROC VM will be created and running.

2. Gather EXTPROC VM system information for DBMS_CLOUD_FUNCTION API input

You must provide EXTPROC VM system information as inputs to the DBMS_CLOUD_FUNCTION APIs in the Autonomous AI Database to enable invocation of external procedures from the database.

- a. From the Oracle Cloud Infrastructure left navigation menu, drop down to Compute, choose the **compartment** you specified for the EXTPROC VM provisioning, and review the dashboard listing all the VMs in the compartment. The ORM Job will have created the EXTPROC VM with a canonical name **extproc-agent**, or **extproc-agent-<setofnumbers>**. This will be the hostname of your EXTPROC VM instance. Clicking on this entry will take you to the page that shows the system details - the IPV4 Public and Private Addresses (IPv4 and IPv6), the subnet, and the image details.
- b. Login to the ExtProc VM using the SSH key that you provided during ExtProc Stack creation, and the IP address. The default user on the VM is `opc`. `sudo` to become user `oracle`.

```
~ ssh -i ssh-key-<date-time-id>.key opc@<Public-IP-Address>
Wed Nov 19 10:46:25 GMT 2025: EXTPROC Agent initialization completed.
```

```
Activate the web console with: systemctl enable --now cockpit.socket
```

```
Last login: Thu Nov 20 20:10:54 2025 from <client-IP-address>
```

```
[opc@extproc-agent-170798 ~]$ whoami
opc
```

```
[opc@extproc-agent-170798 ~]$ sudo su - oracle
Last login: Wed Nov 19 19:07:24 GMT 2025 on pts/0
```

The EXTPROC VM will have a running Podman container with a Oracle SQL*Net listener. Within the container, an Oracle client environment with the necessary configuration files like `sqlnet.ora`, `listener.ora` will be available, along with a wallet file for secure communication with the Autonomous AI Database.

- c. Collect the system information required for the PL/SQL APIs - at the OS level, and next, from the Podman container that is configured in the VM. The container runs the Oracle Listener to receive remote execution requests from the SQL and PL/SQL application on Autonomous AI Database, and is the runtime execution engine for the external procedure.
 - From the Linux OS, the FQDN (fully qualified domain name) of the EXTPROC VM host.
 - From the Podman container, from the listener status, the Port on the EXTPROC VM on which the database connects to the VM.

- From the Podman container, from the the sqlnet.ora specification, the location of the wallet file.
- From an inspection of the wallet file using the orapki tool, the Distinguished Name (DN) parameter value for the wallet certificate.

```
[oracle@extproc-agent-170798 ~]$ hostname -f
extproc-agent-170798.subnet-name.vcn-name.oraclevcn.com
```

```
[oracle@extproc-agent-170798 ~]$ podman ps
CONTAINER ID  IMAGE                                COMMAND
CREATED      STATUS                                PORTS      NAMES
86d81c4df6ff  ghcr.io/oracle/adb-extproc:latest  23 hours  adb-extproc
ago Up 23 hours (healthy)
```

```
[oracle@extproc-agent-170798 ~]$ podman exec -it 86d81c4df6ff bash
```

```
(base) [oracle@extproc-agent-170798 admin]$ lsnrctl status
```

```
LSNRCTL for Linux: Version 23.26.0.0.0 - for Oracle Cloud and
Engineered Systems on 20-NOV-2025 09:25:54
```

Copyright (c) 1991, 2025, Oracle. All rights reserved.

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC1)))
STATUS of the LISTENER
```

```
-----
Alias                LISTENER
Version              TNSLSNR for Linux: Version 23.26.0.0.0 - for
Oracle Cloud and Engineered Systems
Start Date           19-NOV-2025 17:19:58
Uptime               0 days 16 hr. 5 min. 55 sec
Trace Level          off
Security              ON: Local OS Authentication
SNMP                 OFF
Listener Parameter File /u01/app/oracle/product/23.0.0.0/client_1/
network/admin/listener.ora
Listener Log File     /u01/app/oracle/diag/tnslsnr/extproc-
agent-170798/listener/alert/log.xml
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=0.0.0.0)(PORT=16000)))
Services Summary...
Service "extproccontainer.com" has 1 instance(s).
  Instance "extproccontainer", status UNKNOWN, has 1 handler(s) for
this service...
The command completed successfully
```

```
(base) [oracle@extproc-agent-170798 /]$ cd $ORACLE_HOME/network/admin
```

```
(base) [oracle@extproc-agent-170798 admin]$ cat sqlnet.ora
```

```
NAMES.DIRECTORY_PATH= (TNSNAMES, EZCONNECT)
WALLET_LOCATION =(SOURCE = (METHOD = FILE)(METHOD_DATA = (DIRECTORY
= /u01/app/oracle/wallets/extproc_wallet)))
tcp.invited_nodes=(20.63.19.141,20.63.19.141,20.63.19.143,20.63.19.144)
```

```
tcp.validnode_checking=yes
```

```
(base) [oracle@extproc-agent-170798 admin]$ orapki wallet display -
wallet /u01/app/oracle/wallets/extproc_wallet/cwallet.sso
Oracle PKI Tool Release 23.0.0.0.0 - Production
Version 23.0.0.0.0
Copyright (c) 2004, 2025, Oracle and/or its affiliates. All rights
reserved.
```

```
Requested Certificates:
```

```
User Certificates:
```

```
Subject:          CN=extproc-agent-170798
```

```
Trusted Certificates:
```

```
Subject:          CN=extproc-agent-170798
```

```
(base) [oracle@extproc-agent-170798 admin]$ exit
```

```
exit
```

In the above example, we have collected the following inputs for the DBMS_CLOUD_FUNCTION.CREATE_CATALOG API.

- The string formed by FQDN + Port Number will be the input for the `library_listener_url` parameter. In this example, it is 'extproc-agent-170798.subnet-name.vcn-name.oraclevcn.com:16000'.
- `sqlnet.ora` contains the location of the wallet file in the VM: `/u01/app/oracle/wallets/extproc_wallet/cwallet.sso`. Note the IP address list for `tcp.invited_nodes`. It must contain the list of Client IP addresses that you provided as input for network configuration while creating the EXTPROC VM Stack.
- The Certificate Distinguished Name (DN) will be the input for `library_ssl_server_cert_dn` parameter: 'CN=extproc-agent-170798'.

3. Build the C Library and place it in the specified location in the EXTPROC VM Filesystem

Code the C Functions, build it, and place the library in the designated directory in the ExtProc Agent VM's filesystem.

The name of the library must exactly match the name(s) you provided during the creation of the ExtProc Stack. In case you misplaced it, you can find it in the EXTPROC_DLLS entry of the ExtProc container `initextprocontainer.ora` file as shown below, or in the "Variables" section of the ORM (Oracle Resource Manager) Job that created the ExtProc Stack. You must copy the libraries to the `/u01/app/oracle/extproc_libs` directory on the EXTPROC agent VM.

- The full file pathname (in this example `"/u01/app/oracle/extproc_libs/helloCextproc.so"`) will be the input to the parameter `library_remote_path` of `DBMS_CLOUD_FUNCTION.CREATE_CATALOG()`.

```
(base) [oracle@extproc-agent-170798 client_1]$ cat $ORACLE_HOME/hs/admin/
initextprocontainer.ora
SET TRACE_LEVEL=ON
SET _EXTPROC_REMOTE=TRUE
SET WHOAMI=FROMDOCKER
SET LD_LIBRARY_PATH=/u01/app/oracle/product/23.0.0.0/client_1/lib:/u01/app/
oracle/extproc_libs:/opt/conda/lib
SET EXTPROC_DLLS=ONLY:/u01/app/oracle/product/extprocutils.so:/u01/app/
oracle/product/23.0.0.0/client_1/lib/libgsfextproc.so:/u01/app/oracle/
extproc_libs/helloCextproc.so
```

```

SET PYTHONHOME=/opt/conda
SET SCRIPTS_FOLDER_LOC_ENV=/u01/app/oracle/extproc_scripts
SET TRACE_FILE_LOC_ENV=/u01/app/oracle/extproc_logs
SET PYTHONPATH=/tmp:/u01/app/oracle/extproc_scripts
(base) [oracle@extproc-agent-170798 client_1]$ exit

[oracle@extproc-agent-170798 ~]$ cat > helloCextproc.c
#include <stdio.h>
const char* helloCextproc() {
    return ("\nHello C Extproc from FQDN: extproc-agent-170798.subnet-
name.vcn-name.oraclevcn.com\n");
}
^D

[oracle@extproc-agent-170798 ~]$ gcc -shared -fPIC -o /u01/app/oracle/
extproc_libs/helloCextproc.so helloCextproc.c

[oracle@extproc-agent-170798 ~]$ ls -al /u01/app/oracle/extproc_libs/
total 8
drwxr-xr-x. 2 oracle oinstall  30 Nov 19 11:14 .
drwxr-xr-x. 6 root   root      91 Nov 14 02:19 ..
-rwxr-xr-x. 1 oracle oinstall 8184 Nov 19 11:14 helloCextproc.so
[oracle@extproc-agent-170798 ~]$

```

This will complete all the parameter inputs that we require for creating the Catalog/Library object in the database application.

4. Upload the wallet file from the ExtProc VM to OCI Object Storage

A self-signed wallet is created as part of the EXTPROC agent application creation. This wallet allows you to access the Extproc agent instance.

To execute remote procedures at the EXTPROC agent instance, the Autonomous AI Database and the EXTPROC agent connect using Mutual Transport Layer Security (mTLS). When using Mutual Transport Layer Security (mTLS), clients connect through a TCPS (Secure TCP) database connection using standard TLS 1.2 with a trusted client certificate authority (CA) certificate.

Note

You can also obtain and use a public certificate issued by a Certificate Authority (CA).

As a prerequisite, you must export the wallet from the /u01/app/oracle/extproc_wallet directory on the VM where EXTPROC runs, to OCI Object Storage.

Note

- Secure the wallet file. The wallet file, along with the Database user ID and password provide access to the EXTPROC agent instance. Store wallet files in a secure location and share them only with authorized users.
- Do not rename the wallet file. The wallet file in Object Storage must be named cwallet.sso.

5. Create the catalog, define SQL Functions, and Invoke External Procedures as SQL Functions

From the previous steps, we have all the inputs we need for the DBMS_CLOUD_FUNCTION APIs to execute remote procedures from SQL or PL/SQL database applications.

Next, import the wallet, `cwallet.sso`, containing the certificates for the EXTPROC agent instance from Object Storage into a DIRECTORY in your Autonomous AI Database.

Create a credential to access your Object Storage where you stored the wallet file `cwallet.sso`. See `CREATE_CREDENTIAL` Procedure for information about the username and password parameters for different object storage services.

Create a DIRECTORY object in the database, and download the wallet into the directory using `DBMS_CLOUD.GET_OBJECT` API.

```
SQL> SET DEFINE OFF
```

```
SQL> BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'OCI_CREDENTIAL',
    user_ocid       => '<oci_user_ocid>',
    tenancy_ocid    => '<oci_tenancy_ocid>',
    private_key     => '<API-key-dot-pem-file-contents>',
    fingerprint     => '<fingerprint-created-with-API-key>');
END;
/
```

```
SQL> CREATE DIRECTORY extprocwalletidir AS 'extprocwalletidir';
```

Directory created.

```
SQL> SELECT directory_name, directory_path FROM dba_directories WHERE
directory_name LIKE '%EXTPROC%';
```

DIRECTORY_NAME	DIRECTORY_PATH
EXTPROCWALLETDIR	/u02/data/dbfs/<adbd-name>/42E945D608E16DF9E0630301000AF88D/extprocwalletidir

```
SQL> BEGIN
  DBMS_CLOUD.GET_OBJECT (
    credential_name => 'OCI_CREDENTIAL',
    object_uri      => 'https://objectstorage.us-ashburn-1.oraclecloud.com/p/aN.../n/zr.../b/bucket-name-20260129/o/cwallet.sso',
    directory_name  => 'EXTPROCWALLETDIR'
  );
END;
/
```

Next, create the Library Object in the database that represents the C Library in the ExtProc, with the input parameters you have gathered in the previous step.

```
SQL> BEGIN
  DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
```

```

        library_name          => 'EXTPROC_LIBRARY',
        library_listener_url  => 'extproc-agent-170798.subnet-name.vcn-
name.oraclevcn.com:16000',
        library_wallet_dir_name => 'EXTPROCWALLETDIR',
        library_ssl_server_cert_dn => 'CN=extproc-agent-170798',
        library_remote_path   => '/u01/app/oracle/extproc_libs/
helloCextproc.so'
    );
END;
/
PL/SQL procedure successfully completed.

```

```

SQL> SELECT catalog_name from DBA_CLOUD_FUNCTION_CATALOG WHERE
catalog_name LIKE '%EXTPROC%';

```

```

CATALOG_NAME
-----
EXTPROC_LIBRARY

```

Create the SQL function mapped to the ExtProc C function. See the name of the C function from the program written above.

```

SQL> CREATE OR REPLACE FUNCTION HELLOCEXTPROC RETURN VARCHAR2 AS
LANGUAGE C
LIBRARY EXTPROC_LIBRARY
NAME "helloCextproc";
/

```

Function created.

```

SQL>

```

Invoke the SQL Function. It is a successful run (see the body of the C Function from C routine above).

```

SQL> SELECT HELLOCEXTPROC() FROM dual;

```

```

HELLOCEXTPROC()
-----
-----
Hello C Extproc from FQDN: extproc-agent-170798.subnet-name.vcn-
name.oraclevcn.com

```

Manage Data

About Cloning Autonomous Container Database on Dedicated Exadata Infrastructure

APPLIES TO:  Oracle Public Cloud only

Cloning an Autonomous Container Database (ACD) creates a point-in-time copy of your ACD and the Autonomous AI Databases within it. You can use the cloning feature to quickly set up

an ACD from the latest available timestamp for purposes such as testing, development, or analytics.

Clone Type and Source

The only supported clone type for ACDs is **Full Clone**, which includes the metadata and data from the source container database and Autonomous AI Databases within it.

While cloning an ACD, you may:

- Choose a different Exadata Infrastructure or Autonomous Exadata VM Cluster for the clone database.
- Create the clone ACD in the same region or a region different from the clone source.

After submitting a clone request, the clone database shows up as PROVISIONING until the new ACD is available. You cannot initiate a new clone operation on an ACD that is already being cloned until the ongoing operation completes.

Clone Requirements

When cloning an ACD:

- You must have a different Autonomous Exadata VM Cluster (AVMC) available for the destination of the clone.
- The destination AVMC requires 0.5 OCPUs or 2 ECPUs available for each ADB in the ACD being cloned.
- The target AVMC should have the same major database version as the destination AVMC.
- Your keystore should be accessible and allow-listed from the target AVMC.

Clone Limitations

Note the following limitations with cloning an ACD:

- You cannot clone an ACD to an AVMC if there are any Autonomous AI Databases inside the source ACD that already exist in the AVMC. There cannot be duplicate Autonomous AI Databases.
- Access control lists and auto scaling are disabled in the ACD clone.
- You can only clone an ACD from the latest available backup, either local or remote.
- While cloning an ACD from its Details page, its latest available backup will be considered as the clone source.
- While cloning an ACD from its backup, the latest available remote backup will be considered as the clone source.
- The cloned Autonomous AI Databases will each have the minimum allowed ECPUs/OCPUs allocated to them.

Step-by-Step Guides

To learn how to clone an ACD using the OCI console, see [Clone an Autonomous Container Database](#).

You can also use the `CreateAutonomousContainerDatabase` API to clone a database. For information about using the API and signing requests, see [REST APIs](#) and [Security Credentials](#). For information about SDKs, see [Software Development Kits and Command Line Interface](#).

About Cloning Autonomous AI Database on Dedicated Exadata Infrastructure

Cloning is the process of creating a point-in-time copy of your Autonomous AI Database or its backup set. You can use the cloning feature to quickly set up an Autonomous AI Database with historical data for purposes such as testing, development, or analytics.

✓ Tip

The speed of the clone operation depends on the number of CPUs you specify for the clone you are creating. Therefore, you can improve the speed of the clone operation by specifying more CPUs for the clone and then scaling it down to the desired number of CPUs (as described in [Remove CPU or Storage Resources from Autonomous AI Database on Dedicated Exadata Infrastructure](#)) after the clone operation completes.

Clone Types

Autonomous AI Database supports the following clone types:

- **Full Clone:** A full clone creates a new database that includes the metadata and data from the source database.
- **Metadata Clone:** This clone type creates a new database that includes all source database schema metadata, but not the source database data.

Clone Sources

You can create a database clone from any of the following sources:

1. **A running database instance:** You can create a new database instance by cloning an Autonomous AI Database instance.

While cloning a database instance, **you may:**

- Choose a different Exadata Infrastructure, Autonomous Exadata VM Cluster, or Autonomous Container Database for the clone database.
- Create the clone database in the same region or a region different from the clone source.
- Create the clone database in the same tenancy or a tenancy different from the clone source. A cross-tenancy clone can be in the same region or a region different from the clone source. Cross tenancy cloning is supported on Oracle Public Cloud deployments only.

2. **A backup of a database instance:** You can create a new database instance by cloning an automatic backup of the Autonomous AI Database, either an on-demand backup or a long-term backup.

In an Autonomous Data Guard setup, you can clone from a backup at the primary or standby location.

While creating a database instance from backup, **you may:**

- Select a backup from a list of backups within a date range or create a point-in-time clone. Point-in-time clones contain all data up to a specified timestamp. The specified

timestamp must be within the retention period defined at the Autonomous Container Database level.

Note

You can not clone a long-term backup using the Point-in-time clone option. Long-term backups are manual backups that can be retained for a minimum of 90 days and a maximum of 10 years. Refer to About Backup and Recovery for more details.

- Choose a different Exadata Infrastructure, Autonomous Exadata VM Cluster, or Autonomous Container Database for the clone database.
- Create the clone database in the same region or a region different from the clone source.
- Create the clone database in the same tenancy or a tenancy different from the clone source. A cross-tenancy clone can be in the same region or a region different from the clone source. Cross tenancy cloning is supported on Oracle Public Cloud deployments only.

After submitting a clone request, the clone database shows up as PROVISIONING until the new dedicated database is available. You cannot initiate a new clone operation on a dedicated database that is already being cloned until the ongoing operation completes.

Also, note the following information about the newly cloned 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* in Oracle Database 19c SQL Tuning Guide or [Oracle Database 26ai SQL Tuning Guide](#).

- 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.

Clone Requirements

To clone an Autonomous AI Database instance or its backup set successfully, the following requirements have to be met:

- To clone an Autonomous AI Database, you need the required access using the following **policy** statements written by an administrator, whether you're using the Console or the REST API with an SDK, CLI, or another tool:

```
Allow group <Group_Name>
to manage autonomous-databases
in compartment <Compartment_Name>
```

```
Allow group <Group_Name>
to read autonomous-container-databases
in compartment <Compartment_Name>
```

✓ **Tip**

If you try to perform an action and get a message that you don't have permission or are unauthorized, confirm with your administrator the type of access you've been granted and which **compartment** you should work in.

- The target Autonomous Container Database (ACD) must be on the same or higher database version as the source.
- To support cloning on Exadata Cloud@Customer deployments, you must open Port 1522 as part of your network setup. See [Create Autonomous Exadata VM Cluster](#) for details.
- When cloning from a database instance:
 - The source and target encryption key must be the same keystore type.
 - The ADMIN password you specify for the clone database must be different from that of the ADMIN database user in the source database; otherwise, the clone operation will fail.
 - For a Full Clone, the minimum storage that you can specify for the clone database is the source database's actual used space rounded up to the next GB.
- When cloning from a backup:
 - You need a *minimum* of **4 ECPUs** or **1 OCPU** in the target Autonomous Exadata VM Cluster. You can view the number of available CPUs from the Autonomous Exadata VM Clusters listing on the Oracle Cloud Infrastructure console. See [View a List of Autonomous Exadata VM Clusters](#) for more details.
 - The source and target can be different keystore types for the encryption key. However, the following requirements must be met:
 - If both the source and target use customer managed keys using Oracle Key Vault (OKV), they must use the same OKV destination. The target Autonomous Exadata VM Cluster and Autonomous Container Database will require access to the source Oracle Key Vault (OKV) for the keys.
 - On Oracle Cloud, if the source uses customer managed keys via KMS, you must ensure that the target Autonomous Exadata VM Cluster has access to the source KMS vault during the restore operation.

Cross Tenancy Clone Requirements

APPLIES TO:  Oracle Public Cloud only

To create a cross tenancy clone from an Autonomous AI Database instance or its backup set successfully, you must ensure to meet the following requirements:

Note

The cross-tenancy clone requirements discussed below are needed in addition to the general clone requirements discussed in [Clone Requirements](#).

- Run the CLI or API commands to create the cross tenancy clone from the destination tenancy.
- Define OCI Identity and Access Management groups and policies on the source and destination tenancies so that you can run commands to create a clone on the destination tenancy and allow the destination tenancy to contact the source tenancy where the clone source resides. When these policies are revoked, cross tenancy cloning will not be allowed.
 - On the **destination tenancy**, create a group (for example: *DestinationGroup*), and add the user(s) who will be allowed to create the cross tenancy clone to this group. See [Using the Console to Create a Group](#) for guidance.
 - On the **source tenancy**, create IAM policies to allow the group created in the destination tenancy (*DestinationGroup*) to create a clone using a clone source from the source tenancy. See [Using the Console to Create a Policy](#) for guidance.

For example, you can define a policy to allow a user in the *DestinationGroup* of the *DestinationTenancy* read from a specific Autonomous AI Database instance in the specified compartment on the source tenancy as shown below:

```
define tenancy DestinationTenancy as ocid1.tenancy.oc1..unique_ID
define group DestinationGroup as ocid1.group.region1..unique_ID
admit group DestinationGroup of tenancy DestinationTenancy to read
autonomous-database-family
    in compartment ocid1.compartment.region1..unique_ID
    where target.id = 'oc1.autonomousdatabase.oc1..unique_ID'
```

Note

The policy only needs to allow read access on the source Autonomous AI Database instance to create a cross tenancy clone.

The above policy specifies the following:

- * Line 1: OCID of the destination tenancy where you are going to create the clone.
- * Line 2: OCID of the destination group to which the user who will create the clone belongs.
- * Line 3: OCID of the compartment where the clone source resides and the OCID of the clone source (Autonomous AI Database instance or a backup).

Note

The where clause in the above example is optional. It provides a more fine-grained way to grant access to a specific clone source.

- On the **destination tenancy**, create IAM policies to endorse a group to manage the clone source on the source tenancy. See [Using the Console to Create a Policy](#) for guidance.

For example:

```
Define tenancy SourceTenancy as ocid1.tenancy.oc1..unique_ID
Endorse group DestinationGroup to manage autonomous-database-family in
tenancy SourceTenancy
```

The above policy specifies the following:

- * Line 1: OCID of the source tenancy OCID where the clone source resides.
- * Line 2: Specifies the destination group that can be allowed to manage Autonomous AI Databases in the source tenancy.

This policy discussed in the above example allows DestinationGroup to create Autonomous AI Databases and Autonomous AI Database clones in the source tenancy. You can limit cloning permissions so that the group can only clone Autonomous AI Databases but cannot create Autonomous AI Databases, or further limit permission to only create a particular type of clone: Full Clone or Metadata Clone. See [IAM Permissions and API Operations for Autonomous AI Database](#) for more information and examples.

Clone Limitations

There are a few limitations with cloning Autonomous AI Database as listed below:

- You can clone an OCPU database into either OCPU or ECPU database. However, you can not clone an ECPU database into an OCPU database.
- You can not clone an Autonomous AI Database with 23ai version into an Autonomous AI Database with 19c version and vice-versa.
- When cloning from a database instance:
 - For databases using Autonomous Data Guard, you can only clone a primary database. However, you can clone either the primary or standby database when cloning from a backup.
 - You can clone a regular database into an Autonomous AI Database for Developers instance and vice versa. However, to successfully clone a regular database into a developer database, the actual used space of the source database, rounded up to the next GB, must be *32GB or less*.
- When cloning from a backup:
 - Metadata Clone is not supported. You can only use the **Full Clone** option to create a database clone.
 - You can have only one restore operation running in the target Autonomous Exadata VM Cluster at a given time. In other words, you can not have multiple backup clones created on a single Autonomous Exadata VM Cluster simultaneously.
 - You can clone a backup to an Autonomous AI Database for Developers only if the source database's allocated space is *32GB or less*.
 - You can not clone a long-term backup using the Point-in-time clone option.
 - You can resize the CPU to a fractional value only after the clone, if needed. Refer to CPU Overprovisioning to learn more about using fractional CPU values.

- On Exadata Cloud@Customer:
 - * You can not use local disk-based backups for cloning.
 - * The time taken to clone an Autonomous AI Database depends on the CPU Count and the network bandwidth between the Backup Destination and the target Autonomous Container Database.
- Cross tenancy clones:
 - Can only be created using the CLI or the Autonomous AI Database REST APIs. This option is not available using the Oracle Cloud Infrastructure Console.
 - Are only supported on Oracle Public Cloud deployments only.
 - Are not supported with customer managed keys on the source. See Master Encryption Keys in Autonomous AI Database for more information on customer managed keys.

Step-by-Step Guides

To learn how to clone an Autonomous AI Database from different clone sources using the OCI console, see:

- Clone an Autonomous AI Database Instance
- Clone Autonomous AI Database from Backup
- Cross Tenancy Clone an Autonomous AI Database
- Cross Tenancy Clone an Autonomous AI Database from Backup

You can also use the [CreateAutonomousDatabase](#) API to clone a database. For information about using the API and signing requests, see [REST APIs](#) and [Security Credentials](#). For information about SDKs, see [Software Development Kits and Command Line Interface](#).

Create and Manage External Table Cache in Autonomous AI Database

On Autonomous AI Database on Dedicated Exadata Infrastructure, you can use the external table cache to cache the data from external tables within your database.

The external table cache is created as a schema object in your database, which is allocated physical space similar to how tables and indexes are stored in data files. When you create an external table cache, a new table is created in your schema, and any space quota limits that are set for your schema also apply to the external table cache.

About External Table Cache in Autonomous AI Database

An external table cache is a storage area in your Autonomous AI Database that stores the data from an external table.

External data is not managed by the database; however, you can use the external tables to query data outside of the database. Queries on external tables will not be as fast as queries on database tables because each time you access the data it needs to be fetched from the external files stored on Object Store.

The external table cache feature enables you to cache frequently accessed data from external tables within your Autonomous AI Database, without requiring modifications to your applications that access the data, thereby providing faster access to external tables.

The following are some of the benefits of using the external table cache:

- **Improved Performance:** Queries are several times faster for your frequently accessed external data - ideal for dashboards, reports, and analytical tools that access the same data regularly.
- **100% Transparent:** The caching mechanism is entirely transparent; applications can benefit from improved speed without requiring any changes to their queries, dashboards, or applications.
- **Lower Cloud Costs:** In a multi-cloud application, caching reduces the need for repeated external data retrievals from remote storage, thereby reducing data egress fees associated with accessing data across regions or clouds.

You can create external table cache for partitioned and non-partitioned tables created on the following file types:

- Parquet
- ORC
- AVRO
- Iceberg Tables

See Query External Data for more information.

Prerequisites

Lists the prerequisites to create the external table cache.

- You must have the EXECUTE privilege on the DBMS_EXT_TABLE_CACHE package. Note that you can only create an external table cache in your own schema and for the external tables that you own.
- You must have an appropriate space quota allocated for your schema to ensure there is sufficient storage capacity for the cache data.
- You must have credentials to access external table files stored on Object Store. You don't need to create credentials if you enable resource principal credentials for accessing Oracle Cloud Infrastructure Object Store.

Limitations

Lists important notes and restrictions about using external table cache in an Autonomous AI Database.

- You can only create an external table cache in your own schema and for the external tables that you own.
- An external table cache can contain files only from a single external table.
- You cannot grant privileges on an external table cache, and you can only access data from the external table cache that you own.
- When a file is deleted from Object Store, the corresponding data in the cache is flagged as invalid and cannot be accessed.

Create External Table Cache

Run `DBMS_EXT_TABLE_CACHE.CREATE_CACHE` to create an external table cache.

When the cache is created, it is initially empty and enabled for population. The cache size increases each time a file is added, depending on the defined space quota limits for the

schema, until it reaches the assigned limits. See [Set Optional Sizing Preferences for External Table Cache](#) for more information.

1. Use `DBMS_EXT_TABLE_CACHE.CREATE_CACHE` to create external table cache for your schema. For example:

```
BEGIN
  DBMS_EXT_TABLE_CACHE.CREATE_CACHE (
    owner          => 'SALES',
    table_name     => 'STORE_SALES',
    partition_type =>
'PATH');
END;
/
```

This creates a cache for the `STORE_SALES` table in the `SALES` schema. The `STORE_SALES` is an external table pointing to data stored on Object Store.

The `owner` parameter specifies the schema name. This example creates an external table cache for the `SALES` user.

The `partition_type` parameter specifies the partitioning method to use for the external table cache. This example partitions the cache using the `FILE$PATH` column. The `FILE$PATH` is an invisible column that specifies the file path text up to the beginning of the object name.

See [External Table Metadata Columns](#) and `CREATE_CACHE` Procedure for more information.

2. When you first create an external table cache, its metadata is stored in the data dictionary, however no space is allocated for the cache data. You can query the `USER_EXTERNAL_TAB_CACHES` view to verify the cache creation.

```
SELECT external_table_name, cached, disabled
FROM user_external_tab_caches;
```

See [DBA_EXTERNAL_TAB_CACHES and USER_EXTERNAL_TAB_CACHES Views](#) for more information.

Use the `DBMS_EXT_TABLE_CACHE.VALIDATE` procedure to validate an external table cache. An error is reported if the referenced external table is not found in the database.

For example:

```
BEGIN
  DBMS_EXT_TABLE_CACHE.VALIDATE (
    owner          => 'SALES',
    table_name     => 'STORE_SALES',
    raise_errors   => TRUE);
END;
/
```

See `VALIDATE` Procedure for more information.

Populate Files into External Table Cache

Shows examples to populate files into the external table cache.

After you create a cache, you can populate files into the cache. Populating files loads the contents of the specified external table files into the cache. You can choose to populate all files from a table, a specific percentage of the table, or specify a filter condition to limit the files you want to populate. For example, you can filter the files based on their names or a date range.

Depending on the space quota allocated for the schema, Oracle attempts to populate files into the cache. If the assigned quota limit is reached, Oracle stops populating files unless the required space is allocated.

The external table cache does not refresh automatically. To update the cache when a file on the Object Store is modified, you must repopulate the file.

When a file is deleted from the Object Store, the corresponding cached data immediately becomes invalid and cannot be retrieved.

Add Table to External Table Cache

Use `DBMS_EXT_TABLE_CACHE.ADD_TABLE` to populate an entire table or a certain percentage of the external table into the cache.

Examples

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_TABLE (
    owner      => 'SALES',
    table_name => 'STORE_SALES');
END;
/
```

This example attempts to populate the `STORE_SALES` table into the cache, skipping any existing files that have already been populated.

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_TABLE (
    owner      => 'SALES',
    table_name => 'STORE_SALES',
    percent_files => 80);
END;
/
```

This example attempts to populate 80% of the `STORE_SALES` table into the cache, skipping any existing files that have already been populated.

The `percent_files` parameter is optional; if you do not specify this parameter, the entire table is populated into the cache.

See `ADD_TABLE` Procedure for more information.

Add Files to External Table Cache

You can use the following procedures to add one or more files to the external table cache:

- `ADD_FILE`: to add a single file into the cache.

- `ADD_BY_LIKE`: to add one or more specified files based on the specified path filters.
- `ADD_LATEST_FILES`: to add one or more files based on the specified time interval.

Examples

Use the `DBMS_EXT_TABLE_CACHE.ADD_FILE` procedure to populate a single file into the external table cache. For example:

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_FILE (
    owner       => 'SALES',
    table_name  => 'STORE_SALES',
    file_url    => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata.parquet'
  );
END;
/
```

This example populates data from the `salesdata.parquet` file into the cache.

This example skips populating the file into the cache if the specified file exists in the cache and has not been modified since the file was last cached.

See `ADD_FILE` Procedure for more information.

Use the `DBMS_EXT_TABLE_CACHE.ADD_BY_LIKE` procedure to populate one or more files into the external table cache. For example:

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_BY_LIKE (
    owner       => 'SALES',
    table_name  => 'STORE_SALES',
    path_filters => ['"https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata.parquet",
                    "https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata1.parquet"']
  );
END;
/
```

This example populates data from the `salesdata.parquet` and `salesdata1.parquet` files into the cache, skipping any existing files that have already been populated.

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_BY_LIKE (
    owner       => 'SALES',
    table_name  => 'STORE_SALES',
    path_filters => ['"https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/sales#_data1.parquet",
                    "https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/
sales#_data2.parquet"'],
    esc_char    => '#',
    force       => TRUE);
```

```
END;  
/
```

This example populates the sales_data1.parquet and sales_data2.parquet files into the cache.

In this example, '#' character is defined as the escape character. The '_' character following '#' is treated as a literal underscore, not as a wildcard matching any single character.

See ADD_BY_LIKE Procedure for more information.

Use the DBMS_EXT_TABLE_CACHE.ADD_LATEST_FILES procedure to populate one or more files based on the last modified date into the external table cache. For example:

```
BEGIN  
  DBMS_EXT_TABLE_CACHE.ADD_LATEST_FILES (  
    owner      => 'SALES',  
    table_name => 'STORE_SALES',  
    since      => INTERVAL '7' DAY,  
    max_files  => 5,  
    force      => TRUE);  
END;  
/
```

The since parameter specifies the time interval; only files modified within the last seven (7) days are eligible to be populated into the cache.

The max_files parameter limits the number of files that can be populated into the cache. This example populates only five (5) files.

The force parameter forces the specified files to be overwritten in the cache even if the files were not modified.

See ADD_LATEST_FILES Procedure for more information.

Query the following data dictionary views to list the files cached in the external table cache:

- [ALL_EXTERNAL_TAB_CACHE_LOCATIONS](#)
- [USER_EXTERNAL_TAB_CACHE_LOCATIONS](#)

Drop Files from External Table Cache

Shows examples to drop files from external table cache.

You can remove all files from the cache, or specify filter conditions to drop one or more files from the cache. For example, you can filter the files by their names or based on a specific time interval.

Clear External Table Cache

Use DBMS_EXT_TABLE_CACHE.CLEAR to drop all files from the external table cache. For example:

```
BEGIN  
  DBMS_EXT_TABLE_CACHE.CLEAR (  
    owner      => 'SALES',  
    table_name => 'STORE_SALES');  
END;
```

```
END;  
/
```

This example drops all files from the `STORE_SALES` cache and deallocates all space used by the removed files.

See `CLEAR` Procedure for more information.

Drop Files from External Table Cache

You can use the following procedures to drop one or more files from the external table cache:

- `DROP_FILE`: to drop a single file from the cache.
- `DROP_BY_LIKE`: to drop one or more files from the cache based on the specified path filters.
- `RETIRE_FILES`: to drop one or more files from the cache based on the specified interval.

Examples

Use `DBMS_EXT_TABLE_CACHE.DROP_FILE` to drop a file from the external table cache. For example:

```
BEGIN  
  DBMS_EXT_TABLE_CACHE.DROP_FILE (  
    owner      => 'SALES',  
    table_name => 'STORE_SALES',  
    file_url   => 'https://swiftobjectstorage.us-  
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata.parquet'  
  );  
END;  
/
```

This example drops the `salesdata.parquet` file from the cache and deallocates all space used by the removed file.

See `DROP_FILE` Procedure for more information.

Use `DBMS_EXT_TABLE_CACHE.DROP_BY_LIKE` to drop one or more files based on the `path_filters` parameter. For example:

```
BEGIN  
  DBMS_EXT_TABLE_CACHE.DROP_BY_LIKE (  
    owner      => 'SALES',  
    table_name => 'STORE_SALES',  
    path_filters => ['"https://swiftobjectstorage.us-  
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata.parquet",  
                    "https://swiftobjectstorage.us-  
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata1.parquet"]'  
  );  
END;  
/
```

This example drops the salesdata.parquet and salesdata1.parquet files from the cache and deallocates all space used by the removed files.

```
BEGIN
  DBMS_EXT_TABLE_CACHE.DROP_BY_LIKE (
    owner      => 'SALES',
    table_name => 'STORE_SALES',
    path_filters => ['https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/sales#_data1.parquet",
                    "https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/
sales#_data1.parquet"]'
  );
END;
/
```

This example drops the sales#_data1 and sales#_data2 files from the cache and deallocates all space used by the removed files.

In this example, the '#' character is defined as the escape character. The '_' character following '#' is treated as a literal underscore, not as a wildcard matching any single character.

See DROP_BY_LIKE Procedure for more information.

Use DBMS_EXT_TABLE_CACHE.RETIRE_FILES to drop one or more files based on the specified interval. For example:

```
BEGIN
  DBMS_EXT_TABLE_CACHE.RETIRE_FILES (
    owner      => 'SALES',
    table_name => 'STORE_SALES',
    before     => INTERVAL '30' DAY);
END;
/
```

This example drops files that are older than thirty (30) days from the cache and deallocates all space used by the removed files.

See RETIRE_FILES Procedure for more information.

The above examples remove one or more files from the cache while retaining the cache. You can load files again into the cache when necessary. See [Populate Files into External Table Cache](#) for more information.

Disable and Enable External Table Cache

Shows examples to disable and enable the external table cache.

Run DBMS_EXT_TABLE_CACHE.DISABLE to disable external table cache from the database. Disabling a cache does not delete data from the cache; instead, the cache is flagged as **DISABLED**, and the optimizer cannot use the cache for query rewrites.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.DISABLE (
```

```
        owner      => 'SALES',  
        table_name =>  
'STORE_SALES');  
  
END;  
/
```

This example disables the STORE_SALES cache.

See DISABLE Procedure for more information.

After you disable an external table cache, use DBMS_EXT_TABLE_CACHE.ENABLE to enable the cache.

Example:

```
BEGIN  
    DBMS_EXT_TABLE_CACHE.ENABLE (  
        owner      => 'SALES',  
        table_name => 'STORE_SALES'  
    );  
END;  
/
```

This example enables the STORE_SALES cache.

See ENABLE Procedure for more information.

Drop External Table Cache

Shows an example to drop the external table cache.

Run DBMS_EXT_TABLE_CACHE.DROP_CACHE to drop an external table cache. The DBMS_EXT_TABLE_CACHE.DROP_CACHE procedure removes the specified external table cache from the database and releases the storage space associated with the cache.

Example:

```
BEGIN  
    DBMS_EXT_TABLE_CACHE.DROP_CACHE (  
        owner      => 'SALES',  
        table_name =>  
'STORE_SALES');  
END;  
/
```

This example drops the STORE_SALES cache from the SALES schema.

Dropping a cache removes its metadata from the data dictionary and deletes all its cached data.

See DROP_CACHE Procedure for more information.

Query the USER_EXTERNAL_TAB_CACHES view to verify that the cache has been dropped. For example:

```
SELECT external_table_name, cached
FROM user_external_tab_caches;
```

See [DBA_EXTERNAL_TAB_CACHES and USER_EXTERNAL_TAB_CACHES Views](#) for more information.

View External Table Cache Information

Autonomous AI Database provides views that allow you to monitor the external table cache.

View	Description
DBA_EXTERNAL_TAB_CACHES and USER_EXTERNAL_TAB_CACHES Views	Provides information either about all external table caches in the database or about a user's external table caches.
ALL_EXTERNAL_TAB_CACHE_LOCATIONS	Provides information about the files in Cloud Storage that are accessible to the current user and belong to cached external tables.
USER_EXTERNAL_TAB_CACHE_LOCATIONS	Provides information about the files in cloud storage that are owned by the current user and belong to cached external tables. This view does not display the OWNER column.

Set Optional Sizing Preferences for External Table Cache

You can use the DBMS_EXT_TABLE_CACHE.SET_USER_PROPERTY procedure to set sizing preferences and space quota limits on the external table cache.

By default, the external table cache is disabled for a user. To enable and create the external table cache use the DBMS_EXT_TABLE_CACHE.CREATE_CACHE procedure. The cache is created in your default schema and inherits any space quota limits defined for your schema. However, you can also use the DBMS_EXT_TABLE_CACHE.SET_USER_PROPERTY procedure to define space quotas for external table cache. You use the PROPERTY_NAME and PROPERTY_VALUE parameters of the DBMS_EXT_TABLE_CACHE.SET_USER_PROPERTY procedure to set the space quota limits.

The PROPERTY_NAME parameter accepts MAX_CACHE_SIZE and MAX_CACHE_PERCENT values. The MAX_CACHE_SIZE property specifies the total external cache size in bytes. The MAX_CACHE_PERCENT property specifies the total external cache size as a percentage of the specified user's quota.

Before you set the cache size properties, note the following precedence order:

- When MAX_CACHE_SIZE, MAX_CACHE_PERCENT, and the space quota are defined, MAX_CACHE_PERCENT takes precedence over MAX_CACHE_SIZE.
- When only MAX_CACHE_SIZE is defined, and MAX_CACHE_PERCENT or space quota is not defined, MAX_CACHE_SIZE takes precedence.
- When only the space quota is defined and MAX_CACHE_SIZE and MAX_CACHE_PERCENT are not defined, the cache size quota defaults to 10% of the total schema quota.
- When MAX_CACHE_SIZE, MAX_CACHE_PERCENT, or the space quota is not defined, the cache space quota defaults to UNLIMITED.

Examples

```
BEGIN
  DBMS_EXT_TABLE_CACHE.SET_USER_PROPERTY (
    property_name => 'MAX_CACHE_PERCENT',
    property_value => 50,
    owner         =>
'SALES');
END;
/
```

This example sets the caching preference to MAX_CACHE_PERCENT for the SALES schema.

The property_value is 50%, which specifies that the cache space quota for the SALES schema is a maximum of 50% of the total space quota defined for SALES.

```
BEGIN
  DBMS_EXT_TABLE_CACHE.SET_USER_PROPERTY (
    owner         => 'SALES',
    property_name => 'MAX_CACHE_SIZE',
    property_value =>
5368709120);
END;
/
```

This example sets the caching preference to MAX_CACHE_SIZE for the SALES schema.

The property_value is 5368709120, which specifies that the maximum cache size for the SALES schema is up to 5GB.

See SET_USER_PROPERTY Procedure and CREATE_CACHE Procedure for more information.

Use DBMS_EXT_TABLE_CACHE.GET_USER_PROPERTY to retrieve the cache size properties.

Example:

```
SET SERVEROUTPUT ON
DECLARE
  max_cache_sz NUMBER,
BEGIN
  max_cache_sz := DBMS_EXT_TABLE_CACHE.GET_USER_PROPERTY (
    property_name => 'MAX_CACHE_SIZE',
    owner         =>
'SALES');
END;
/
```

See GET_USER_PROPERTY Function for more information.

External Table Metadata Columns

The external table metadata helps you determine where data is coming from when you perform a query.

The external tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`, `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`, or `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from.

- `file$path`: Specifies the file path text up to the beginning of the object name.
- `file$name`: Specifies the object name, including all the text that follows the final `"/`.

For example:

```
SELECT genre_id, name, file$name, file$path FROM ext_genre
       WHERE rownum <= 2;
```

genre_id	name	file\$name	file\$path
1	Action	genre.csv	https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespaces-string/b/moviestream_gold/o/genre
2	Adventure	genre.csv	https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespaces-string/b/moviestream_gold/o/genre

See [Invisible Columns](#) for more information on invisible columns.

Use Oracle Workspace Manager in Autonomous AI Database

Use Oracle Workspace Manager to version-enable one or more user tables in the database. When a table is version-enabled, all rows in the table can support multiple versions of the data.

Oracle Workspace Manager is enabled by default on Autonomous AI Database on Dedicated Exadata Infrastructure.

Applications that can benefit from Workspace Manager typically do one or more of the following operations:

- Manage a collection of updates and insertions as a unit before incorporating them into production data
- Support a collaborative development effort
- Use a common data set to create multiple scenarios for what-if analyses or multiple editions of data for publication
- Keep a history of changes to data

[Introduction to Workspace Manager](#) provides relevant usage and reference information.

Link Data

Create TLS Database Links from an Autonomous AI Database to another Autonomous AI Database

You can create TLS database links from an Autonomous AI Database on Dedicated Exadata Infrastructure to a target Autonomous AI Database that is on a public endpoint.

Create One-way TLS Database Links from an Autonomous AI Database to another Autonomous AI Database

You can create one-way TLS database links from an Autonomous AI Database on Dedicated Exadata Infrastructure to a publicly accessible Autonomous AI Database.

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

See [Obtain Tenancy Details](#) for information on finding the outbound IP address.

To create a database link to a target Autonomous AI Database without a wallet (TLS):

1. Ensure that one-way TLS connections are enabled.

By default, one-way TLS connections are enabled when you provision an AVMC. See [Create an Autonomous Exadata VM Cluster](#) for more information.

Note

In an Autonomous AI Database on Dedicated Exadata Infrastructure, walletless connections work for one-way TLS by installing a custom SSL certificate issued by a public CA. For more details, see [Prepare for TLS Walletless Connections](#).

2. On the Autonomous AI Database instance where you are creating the database link, create credentials to access the target Autonomous AI Database. The username and password you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database (you use these credentials to create the database link).

Note

Supplying the `credential_name` parameter is required.

For example:

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

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

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

3. Create a database link to the target Autonomous AI Database instance using `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name => 'SALESLINK',
    hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
    port => '2484',
    service_name => 'example_medium.atpc.example.oraclecloud.com',
    credential_name => 'DB_LINK_CRED',
    directory_name => NULL);
END;
/
```

To create a database link with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to a target Autonomous AI Database on a public endpoint using a secure TCP connection without a wallet, the `directory_name` parameter must be `NULL`.

The `port` parameter value must be set to 2484 for TLS connections. However, if a different SCAN listener port number was chosen while provisioning the AVMC, you must use that port number. See [Create Autonomous Exadata VM Cluster](#) for more information.

The `ssl_server_cert_dn` parameter can either be omitted or if included, specify a `NULL` value.

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

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

For example:

```
SELECT * FROM employees@SALESLINK;
```

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

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

Where *password* is the new password.

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

For additional information, see:

- `CREATE_DATABASE_LINK` Procedure
- `UPDATE_CREDENTIAL` Procedure

Create mTLS Database Links from an Autonomous AI Database to another Autonomous AI Database

You can create mTLS database links from an Autonomous AI Database on Dedicated Exadata Infrastructure to a publicly accessible Autonomous AI Database.

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

See Obtain Tenancy Details for information on finding the outbound IP address.

To create database links to a target Autonomous AI Database with a wallet (mTLS):

1. Ensure that mTLS connections are enabled.

By default, one-way TLS connections are enabled for an AVMC. You can enable mutual TLS (mTLS) authentication by selecting the **Enable mutual TLS (mTLS) authentication** check-box while provisioning the AVMC. See Create an Autonomous Exadata VM Cluster for more information.

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

Note the following for the wallet file:

- The wallet file, along with the Database user ID and password provide access to data in the target Oracle Database. Store wallet files in a secure location. Share wallet files only with authorized users.
- Do not rename the wallet file. The wallet file in Object Storage must be named `cwallet.sso`.

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

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

For example:

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

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

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

For example:

```
BEGIN
  DBMS_CLOUD.GET_OBJECT(
    credential_name => 'DEF_CRED_NAME',
    object_uri => 'https://objectstorage.us-
```

```

phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/cwallet.sso',
    directory_name => 'DBLINK_WALLET_DIR');
END;
/

```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

6. On the Autonomous AI Database instance, create credentials to access the target database. The username and password you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database that you use to create the database link.

Note

Supplying the `credential_name` parameter is required.

For example:

```

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

```

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

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

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

For example:

```

BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'SALES LINK',
        hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
        port => '2484',
        service_name => 'example_medium.atpc.example.oraclecloud.com',
        ssl_server_cert_dn => 'CN=atpc.example.oraclecloud.com, OU=Oracle
BMCS FRANKFURT, O=Oracle Corporation, L=Redwood City, ST=California, C=US',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'DBLINK_WALLET_DIR');
END;
/

```

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

The `port` parameter value must be set to 2484 for TLS connections. However, if a different SCAN listener port number was chosen while provisioning the AVMC, you must use that port number. See [Create Autonomous Exadata VM Cluster](#) for more information.

If the wallet file in the directory specified with `directory_name` is not `cwallet.sso`, the procedure reports an error such as: `ORA-28759: failure to open file`.

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

For example:

```
SELECT * FROM employees@SALES LINK;
```

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

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

Where *password* is the new password.

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

For additional information, see:

- [CREATE_CREDENTIAL Procedure](#)
- [GET_OBJECT Procedure and Function](#)
- [UPDATE_CREDENTIAL Procedure](#)

Database Link Notes with a Target that is an Autonomous AI Database

Provides notes for creating database links to a target that is another Autonomous AI Database.

Notes for database links to another Autonomous AI Database:

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

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

- To list the database links, use the `ALL_DB_LINKS` view. See [ALL_DB_LINKS](#) in Oracle Database 19c Database Reference or [Oracle Database 26ai Database Reference](#) for more information.

- The wallet file, along with the Database user ID and password provide access to data in the target Autonomous AI Database. Store wallet files in a secure location. Share wallet files only with authorized users.
- The `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` input should mention the scan name of the target Dedicated Autonomous AI Database as the 'hostname' parameter.

Create Non-TLS Database Links from Autonomous AI Database to an Autonomous AI Database

You can create Non-TLS database links from an Autonomous AI Database on Dedicated Exadata Infrastructure to a target Autonomous AI Database that is on a private endpoint.

Prerequisites for Non-TLS Database Links from Autonomous AI Database to a Target Autonomous AI Database

Lists the prerequisites to create Non-TLS database links to a target Autonomous AI Database that is on a private endpoint.

To create a database link to a target Autonomous AI Database on a private endpoint:

- The target database must be accessible from the source database's Oracle Cloud Infrastructure VCN. For example, you can connect to the target database when:
 - The target database is on a private endpoint.
 - Both the source database and the target database are in the same Oracle Cloud Infrastructure VCN.
 - The source database and the target database are in different Oracle Cloud Infrastructure VCNs that are paired.
 - The target database is connected to the source database's Oracle Cloud Infrastructure VCN using FastConnect or VPN.
- For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).
- `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` does not support a value of `localhost` for the `hostname` parameter.
- The following ingress and egress rules must be defined for the private endpoint:
 - Define an egress rule in the source database's subnet security list or network security group such that the traffic over TCP is allowed to the target database's IP address and port number.
 - Define an ingress rule in the target database's subnet security list or network security group such that the traffic over TCP is allowed from the source database IP address to the destination port.

Create Non-TLS Database Links to a Target Autonomous AI Database

You can create Non-TLS database links from an Autonomous AI Database on Dedicated Exadata Infrastructure to a target Autonomous AI Database that is on a private endpoint.

Perform the prerequisite steps, as required. See [Prerequisites for Non-TLS Database Links from Autonomous AI Database to a Target Autonomous AI Database](#) for details.

To create a Non-TLS database link to a target Autonomous AI Database on a private endpoint:

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

For example:

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

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

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

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

For example:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'PRIVATE_ENDPOINT_LINK',
        hostname => 'exampleHostname',
        port => '1521',
        service_name => 'example_high.adb.oraclecloud.com',
        credential_name => 'PRIVATE_ENDPOINT_CRED',
        directory_name => NULL,
        private_target => TRUE);
END;
/
```

For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).

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

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

- The `port` parameter value must be set to 1521 for Non-TLS connections. However, if a different SCAN listener port number was chosen while provisioning the AVMC, you must use that port number. See [Create Autonomous Exadata VM Cluster](#) for more information.

The `directory_name` parameter must be NULL.

- The `ssl_server_cert_dn` parameter can either be omitted or if included, specify a NULL value.
- The `private_target` parameter must be TRUE.

Note

Setting the `private_target` parameter to TRUE creates the database link using the TCP protocol instead of TCPS, thereby not validating the SSL certificate during the connection.

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

For example:

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

Note

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

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

Where *password* is the new password.

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

See `CREATE_DATABASE_LINK` Procedure for additional information.

Database Link Notes with a Target that is an Autonomous AI Database

Provides notes for creating database links to a target that is another Autonomous AI Database.

Notes for database links to another Autonomous AI Database:

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

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

- To list the database links, use the `ALL_DB_LINKS` view. See [ALL_DB_LINKS](#) in Oracle Database 19c Database Reference or [Oracle Database 26ai Database Reference](#) for more information.
- The wallet file, along with the Database user ID and password provide access to data in the target Autonomous AI Database. Store wallet files in a secure location. Share wallet files only with authorized users.
- The `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` input should mention the scan name of the target Dedicated Autonomous AI Database as the 'hostname' parameter.

Create Database Links from an Autonomous AI Database to a Publicly Accessible Oracle Database with a Wallet (mTLS)

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

To use database links with Autonomous AI Database the target database must be configured to use TCP/IP with SSL (TCPS) authentication. Autonomous AI Databases use TCP/IP with SSL (TCPS) authentication by default, so you do not need to do any additional configuration in your target database to link to another Autonomous AI Database. Other Oracle databases must be configured to use TCP/IP with SSL (TCPS) authentication. See [Configuring Secure Sockets Layer Authentication](#) in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for more information.

To create database links to a public target, the target Oracle Database must be accessible. Some databases may limit access (for example, using Access Control Lists). Make sure you enable your target database to allow access from your source database for the database link to work. If you limit access with Access Control Lists (ACLs), you can find the outbound IP address of your source Autonomous AI Database and allow that IP address to connect to your target database.

See [How to Create a Database Link from Your Autonomous AI Database to a Database Cloud Service Instance](#) for more information.

To create database links to a target Oracle database with a wallet (mTLS):

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

Note the following for the wallet file:

- The wallet file, along with the Database user ID and password provide access to data in the target Oracle Database. Store wallet files in a secure location. Share wallet files only with authorized users.
 - Do not rename the wallet file. The wallet file in Object Storage must be named `cwallet.sso`.
2. Create credentials to access your Object Store where you store the wallet file `cwallet.sso`. See [CREATE_CREDENTIAL Procedure](#) for information about the username and password parameters for different object storage services.
 3. Create a directory on Autonomous AI Database for the wallet file `cwallet.sso`.

For example:

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

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

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

For example:

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

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

Note

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

5. On the Autonomous AI Database instance, create credentials to access the target database. The username and password you specify with `DBMS_CLOUD.CREATE_CREDENTIAL` are the credentials for the target database that you use to create the database link.

Note

Supplying the `credential_name` parameter is required.

For example:

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

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

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

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

For example:

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

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

If the wallet file in the directory specified with `directory_name` is not `cwallet.sso`, the procedure reports an error such as: `ORA-28759: failure to open file`.

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

For example:

```
SELECT * FROM employees@SALESLINK;
```

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

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

Where *password* is the new password.

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

For additional information, see:

- CREATE_DATABASE_LINK Procedure
- GET_OBJECT Procedure and Function
- UPDATE_CREDENTIAL Procedure

Database Link Notes with a Target Oracle Database

Provides notes for creating database links to a target Oracle database (when the target is not an Autonomous AI Database)

Notes for database links to other Oracle databases:

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

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

- Supported target Oracle database versions for database links to another Oracle Database are: 19c, 12.2.0, and 12.1.0.

Note

For complete information on supported versions, see [Client Server Interoperability Support Matrix for Different Oracle Versions \(Doc ID 207303.1\)](#)

- Autonomous AI Database sets the `SEC_CASE_SENSITIVE_LOGON` parameter to true and this value cannot be changed. If your target database is not an Autonomous AI Database, then you must set `SEC_CASE_SENSITIVE_LOGON` parameter to true on the target database. If `SEC_CASE_SENSITIVE_LOGON` is set to false on the target database, then error ORA-28040: No matching authentication protocol is raised.
- To list the database links, use the `ALL_DB_LINKS` view. See [ALL_DB_LINKS](#) in Oracle Database 19c Database Reference or [Oracle Database 26ai Database Reference](#) for more information.
- The wallet file, along with the Database user ID and password provide access to data in the target Oracle database. Store wallet files in a secure location. Share wallet files only with authorized users.
- When the Autonomous AI Database instance is on a private endpoint, there are two options to specify the target database: use either the `hostname` parameter or the `rac_hostnames` parameter:
 - For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).
 - When the target is an Oracle RAC database, use the `rac_hostnames` parameter to specify one or more hostnames with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the `rac_hostnames` value is not supported.

When you specify a list of host names in the `rac_hostnames` parameter, `CREATE_DATABASE_LINK` uses all of the specified host names as addresses in the connect string. If one of the specified hosts is not available on the target Oracle RAC database, Autonomous Database automatically attempts to connect using another host name from the list.

 - `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` does not support a value of `localhost` for the `hostname` or in the `rac_hostnames` parameter.

Create Database Links from Autonomous AI Database to an Oracle Database on a Private Endpoint

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

Note

Database links from an Autonomous AI Database to a target Oracle database that is on a private endpoint are only supported in commercial regions and US Government regions.

This feature is enabled by default in all commercial regions.

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

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

US Government regions include the following:

- [Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization](#)
- [Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization](#)

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

- **Another Oracle Database, such as on-premises or a Database Cloud Service database, on a private endpoint that is configured for SSL (TCPS):** In this case you can create the database link with a wallet, and the database link communicates with TCPS. See [Create Database Links from Autonomous AI Database to Oracle Databases on a Private Endpoint with a Wallet \(mTLS\)](#) for details.
- **Oracle Database, such as on-premises or a Database Cloud Service database, on a private endpoint that is configured for TCP:** In this case you create the database link without a wallet and the database link communicates with TCP. See [Create Database Links to Oracle Databases on a Private Endpoint without a Wallet](#) for details.

Prerequisites for Database Links from Autonomous AI Database to Oracle Databases on a Private Endpoint

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

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

- The target database must be accessible from the source database's Oracle Cloud Infrastructure VCN. For example, you can connect to the target database when:
 - The target database is on a private endpoint.
 - Both the source database and the target database are in the same Oracle Cloud Infrastructure VCN.

- The source database and the target database are in different Oracle Cloud Infrastructure VCNs that are paired.
 - The target database is an on-premises database that is connected to the source database's Oracle Cloud Infrastructure VCN using FastConnect or VPN.
 - There are two options to specify the target database, use the `hostname` parameter or the `rac_hostnames` parameter:
 - For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).
 - When the target is an Oracle RAC database, use the `rac_hostnames` parameter to specify one or more hostnames with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the `rac_hostnames` value is not supported.
- When you specify a list of host names in the `rac_hostnames` parameter, `CREATE_DATABASE_LINK` uses all of the specified host names as addresses in the connect string. If one of the specified hosts is not available on the target Oracle RAC database, Autonomous Database automatically attempts to connect using another host name from the list.
- The following ingress and egress rules must be defined for the private endpoint:
 - Define an egress rule in the source database's subnet security list or network security group such that the traffic over TCP is allowed to the target database's IP address and port number.
 - Define an ingress rule in the target database's subnet security list or network security group such that the traffic over TCP is allowed from the source database IP address to the destination port.

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

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

Note

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

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

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

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

For example:

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

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

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

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

For example:

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

For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).

When the target is an Oracle RAC database, use the `rac_hostnames` parameter to specify one or more hostnames with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the `rac_hostnames` value is not supported.

For example, with a target Oracle RAC database use the `rac_hostnames` parameter:

```
BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'PRIVATE_ENDPOINT_LINK',
        rac_hostnames => '['sales1-svr1.example.adb.us-
ashburn-1.oraclecloud.com",
                        "sales1-svr2.example.adb.us-
ashburn-1.oraclecloud.com",
                        "sales1-svr3.example.adb.us-
ashburn-1.oraclecloud.com"]',
        port => '1522',
```



```

        service_name => 'exampleServiceName',
        ssl_server_cert_dn => NULL,
        credential_name => 'PRIVATE_ENDPOINT_CRED',
        directory_name => NULL,
        private_target => TRUE);
END;
/

```

DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK does not support a value of localhost for the hostname or in the rac_hostnames parameter.

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

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

- The directory_name parameter must be NULL.
 - The ssl_server_cert_dn parameter must be NULL.
 - The private_target parameter must be TRUE.
3. Use the database link you created to access data in the target database.

For example:

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

Note

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

```

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

```

Where *password* is the new password.

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

See CREATE_DATABASE_LINK Procedure for additional information.

Create Database Links from Autonomous AI Database to Oracle Databases on a Private Endpoint with a Wallet (mTLS)

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

① Note

This option is for target Oracle databases that have SSL/TCPs configured and that are on a private endpoint.

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

- You can configure the target Oracle database to use TCP/IP with SSL (TCPS) authentication. See *Configuring Transport Layer Security Authentication* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for information on configuring SSL/TCPs.
- You can connect to the target Oracle database with TCP. See [Create Database Links to Oracle Databases on a Private Endpoint without a Wallet](#) for details.

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

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

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

① Note

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

2. Create credentials to access your Object Store where you store the `cwallet.sso`. See `CREATE_CREDENTIAL` Procedure for information about the username and password parameters for different object storage services.
3. Create a directory on Autonomous AI Database for the wallet file `cwallet.sso`.

For example:

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

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

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

For example:

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

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

Note

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

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

Note

Supplying the `credential_name` parameter is required.

For example:

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

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

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

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

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name => 'PEDBLINK1',
    hostname => 'example1.adb.ap-osaka-1.oraclecloud.com',
```

```

        port => '1522',
        service_name => 'example_high.adb.oraclecloud.com',
        ssl_server_cert_dn => 'ssl_server_cert_dn',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'WALLET_DIR',
        private_target => TRUE);
END;
/

```

For a target on a private endpoint, DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK supports specifying a single hostname with the hostname parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, CREATE_DATABASE_LINK supports using an IP address, a SCAN IP, or a SCAN hostname).

When the target is an Oracle RAC database, use the rac_hostnames parameter to specify one or more hostnames with DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the rac_hostnames value is not supported.

For example, with a target Oracle RAC database use the rac_hostnames parameter:

```

BEGIN
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'PEDBLINK1',
        rac_hostnames => '["sales1-svr1.example.adb.us-
ashburn-1.oraclecloud.com",
                        "sales1-svr2.example.adb.us-
ashburn-1.oraclecloud.com",
                        "sales1-svr3.example.adb.us-
ashburn-1.oraclecloud.com"]',
        port => '1522',
        service_name => 'example_high.adb.oraclecloud.com',
        ssl_server_cert_dn => 'ssl_server_cert_dn',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'WALLET_DIR',
        private_target => TRUE);
END;
/

```

DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK does not support a value of localhost for the hostname or in the rac_hostnames parameter.

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

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

For example:

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

Note

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

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

Where *password* is the new password.

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

See CREATE_DATABASE_LINK Procedure for additional information.

Database Link Notes with a Target Oracle Database

Provides notes for creating database links to a target Oracle database (when the target is not an Autonomous AI Database)

Notes for database links to other Oracle databases:

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

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

- Supported target Oracle database versions for database links to another Oracle Database are: 19c, 12.2.0, and 12.1.0.

Note

For complete information on supported versions, see [Client Server Interoperability Support Matrix for Different Oracle Versions \(Doc ID 207303.1\)](#)

- Autonomous AI Database sets the `SEC_CASE_SENSITIVE_LOGON` parameter to true and this value cannot be changed. If your target database is not an Autonomous AI Database, then you must set `SEC_CASE_SENSITIVE_LOGON` parameter to true on the target database. If `SEC_CASE_SENSITIVE_LOGON` is set to false on the target database, then error ORA-28040: No matching authentication protocol is raised.

- To list the database links, use the `ALL_DB_LINKS` view. See *ALL_DB_LINKS* in Oracle Database 19c Database Reference or [Oracle Database 26ai Database Reference](#) for more information.
- The wallet file, along with the Database user ID and password provide access to data in the target Oracle database. Store wallet files in a secure location. Share wallet files only with authorized users.
- When the Autonomous AI Database instance is on a private endpoint, there are two options to specify the target database: use either the `hostname` parameter or the `rac_hostnames` parameter:
 - For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, `SCAN IP`, or a `SCAN` hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a `SCAN IP`, or a `SCAN` hostname).
 - When the target is an Oracle RAC database, use the `rac_hostnames` parameter to specify one or more hostnames with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a `SCAN IP`, or a `SCAN` hostname in the `rac_hostnames` value is not supported.

When you specify a list of host names in the `rac_hostnames` parameter, `CREATE_DATABASE_LINK` uses all of the specified host names as addresses in the connect string. If one of the specified hosts is not available on the target Oracle RAC database, Autonomous Database automatically attempts to connect using another host name from the list.
 - `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` does not support a value of `localhost` for the `hostname` or in the `rac_hostnames` parameter.

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

Use `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` to create database links from an Autonomous AI Database on Dedicated Exadata Infrastructure that is on a public endpoint to an Oracle Database Gateway to access Non-Oracle databases.

An Oracle Database Gateway is a gateway that is designed for accessing a specific non-Oracle system. Using an Oracle Database Gateway, you can access data anywhere in a distributed database system without knowing either the location of the data or how it is stored. Using database links on Autonomous AI Database with Oracle Database Gateway supports heterogeneous environments and eliminates the need to customize your applications to access data from non-Oracle systems.

Note

Creating database links with customer-managed heterogeneous connectivity to publicly accessible non-Oracle databases is supported only with versions 19.25 and above for 19c and 23.6 and above for 23ai.

Prerequisites

To use database links from an Autonomous AI Database instance on a public endpoint, do the following:

1. Configure the Oracle Database Gateway to access a non-Oracle database. See *Oracle Database Gateways* in Oracle Database 19c Database Heterogeneous Connectivity User's Guide or [Oracle Database 26ai Database Heterogeneous Connectivity User's Guide](#) for more details.

Depending on the database you want to connect to, you may refer to the corresponding Installation and Configuration Guide and the Gateway User's Guide.

For example, for Oracle Database Gateway for SQL Server see:

- *Installing and Configuring Oracle Database Gateway for SQL Server* in Oracle Database 19c Installation and Configuration Guide for Microsoft Windows or [Oracle Database 26ai Installation and Configuration Guide for Microsoft Windows](#)
 - *Introduction to the Oracle Database Gateway for SQL Server* in Oracle Database 19c User's Guide or [Oracle Database 26ai User's Guide](#)
 - *Configure Oracle Net for the Gateway* in Oracle Database 19c Installation and Configuration Guide for Microsoft Windows or [Oracle Database 26ai Installation and Configuration Guide for Microsoft Windows](#)
2. Configure Oracle Net Listener to handle incoming requests on the Oracle Database Gateway.
 3. Create a self signed wallet on the Oracle Database Gateway.
 4. Configure the target gateway to use TCP/IP with SSL (TCPS) authentication. See *Configuring Transport Layer Security Authentication* in Oracle Database 19c Security Guide or [Oracle Database 26ai Security Guide](#) for more information.

Procedure

Use DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK to create database links from an Autonomous AI Database instance on a public endpoint to an Oracle Database Gateway to access Non-Oracle databases.

To create database links from an Autonomous AI Database instance on a public endpoint to a target gateway:

1. Copy the target gateway self signed wallet, for example, cwallet.sso, containing the certificates for the Oracle Database Gateway to Object Store.

Note

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

2. Create credentials to access the Object Store where you store the cwallet.sso. See CREATE_CREDENTIAL procedure for information about the username and password parameters for different object storage services.
3. Create a directory on Autonomous AI Database for the wallet file cwallet.sso.

For example:

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

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

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

For example:

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

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

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

See [GET_OBJECT Procedure and Function](#) for more details.

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

For example:

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

Supplying the `credential_name` parameter is mandatory.

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

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

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

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name => 'SALESLINK',
```



```
        hostname =>      'example.com',
        port =>          '1522',
        service_name =>   'example_service_name',
        ssl_server_cert_dn => 'ssl_server_cert_dn',
        credential_name => 'DB_LINK_CRED',
        directory_name => 'DBLINK_WALLET_DIR',
        gateway_link =>   TRUE
    );
END;/
```

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

See CREATE_DATABASE_LINK Procedure for more information.

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

For example:

```
SELECT * FROM employees@SALES LINK;
```

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

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

Where password is the new password.

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

See UPDATE_CREDENTIAL Procedure for more details.

See [How to Access Non-Oracle Databases from Autonomous AI Database using Oracle Database Gateway](#) for an example to create a database link to an Oracle Database Gateway to access a Microsoft SQL Server database.

Note

Though the above blog discusses in the context of Autonomous AI Database Serverless, all the content in that blog applies to Autonomous AI Database on Dedicated Exadata Infrastructure also.

Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint

You can create database links from an Autonomous AI Database on Dedicated Exadata Infrastructure to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint.

An Oracle Database Gateway is a gateway that is designed for accessing a specific non-Oracle system. Using an Oracle Database Gateway, you can access data anywhere in a distributed database system without knowing either the location of the data or how it is stored. Using database links on Autonomous AI Database with Oracle Database Gateway supports heterogeneous environments and eliminates the need to customize your applications to access data from non-Oracle systems.

Note

Creating database links with customer-managed heterogeneous connectivity to non-Oracle databases on a private endpoint is supported only with versions 19.25 and above for 19c and 23.6 and above for 23ai.

Prerequisites

To use database links from an Autonomous AI Database instance on a public endpoint, do the following:

1. Configure the Oracle Database Gateway to access a non-Oracle database. See *Oracle Database Gateways* in Oracle Database 19c Database Heterogeneous Connectivity User's Guide or [Oracle Database 26ai Database Heterogeneous Connectivity User's Guide](#) for more details.

Depending on the database you want to connect to, you may refer to the corresponding Installation and Configuration Guide and the Gateway User's Guide.

For example, for Oracle Database Gateway for SQL Server see:

- *Installing and Configuring Oracle Database Gateway for SQL Server* in Oracle Database 19c Installation and Configuration Guide for Microsoft Windows or [Oracle Database 26ai Installation and Configuration Guide for Microsoft Windows](#)
 - *Introduction to the Oracle Database Gateway for SQL Server* in Oracle Database 19c User's Guide or [Oracle Database 26ai User's Guide](#) in *Oracle Database Gateway for SQL Server User's Guide*.
 - *Configure Oracle Net for the Gateway* in Oracle Database 19c Installation and Configuration Guide for Microsoft Windows or [Oracle Database 26ai Installation and Configuration Guide for Microsoft Windows](#)
2. Configure Oracle Net Listener to handle incoming requests on the Oracle Database Gateway.
 3. Create a self signed wallet on the Oracle Database Gateway.

Additionally:

- The target database must be accessible from the source database's Oracle Cloud Infrastructure VCN. For example, you can connect to the target database when:
 - The target database is on a private endpoint.

- Both the source database and the target database are in the same Oracle Cloud Infrastructure VCN.
- The source database and the target database are in different Oracle Cloud Infrastructure VCNs that are paired.
- For a target on a private endpoint, `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` supports specifying a single hostname with the `hostname` parameter. On a private endpoint, using an IP address, SCAN IP, or a SCAN hostname is not supported (when the target is on a public endpoint, `CREATE_DATABASE_LINK` supports using an IP address, a SCAN IP, or a SCAN hostname).
- The following ingress and egress rules must be defined for the private endpoint:
 - Define an egress rule in the source database's subnet security list or network security group such that the traffic over TCP is allowed to the target database's IP address and port number.
 - Define an ingress rule in the target database's subnet security list or network security group such that the traffic over TCP is allowed from the source database IP address to the destination port.

Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint (without a wallet)

You can create database links from an Autonomous AI Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint either with a wallet (TCPS), or without a wallet (TLS). This section describes creating a database link without a wallet.

To create database links from an Autonomous AI Database instance on a private endpoint to a target gateway, using database links and a TLS connection, do the following:

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

For example:

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

Supplying the `credential_name` parameter is mandatory.

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

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

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

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name =>      'SALESLINK',
    hostname =>          'example.com',
    port =>              '1522',
    service_name =>      'example_service_name',
    ssl_server_cert_dn => 'ssl_server_cert_dn',
    credential_name =>    'DB_LINK_CRED',
    directory_name =>     NULL,
    gateway_link =>       TRUE,
    private_target =>     TRUE,
    gateway_params =>     NULL
  );
END;/
```

The `ssl_server_cert_dn` parameter is optional if the connection is created as a TCP based database link (without a wallet).

When the `directory_name` is NULL, the connection is created as a TCP based database link (without a wallet).

The `private_target` parameter must be set to TRUE when the target non-Oracle Database is on a private endpoint (that is, the database link accesses a hostname that needs to be resolved in a VCN DNS server). When `private_target` is TRUE, the `hostname` parameter must be a single hostname (on a private endpoint, using an IP address, a SCAN IP, or a SCAN hostname is not supported).

When `gateway_link` is TRUE and `gateway_params` is NULL, this specifies that the database link is to a customer-managed Oracle gateway.

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

See CREATE_DATABASE_LINK Procedure for more details.

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

For example:

```
SELECT * FROM employees@SALESLINK;
```

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

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

Where `password` is the new password.

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

See UPDATE_CREDENTIAL Procedure for more details.

Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint (with a wallet)

You can create database links from an Autonomous AI Database to an Oracle Database Gateway to access Non-Oracle databases that are on a private endpoint either with a wallet (TCPS), or without a wallet (TLS). This section describes creating a database link with a wallet.

To create database links from an Autonomous AI Database instance on a private endpoint to a target gateway, connecting with a wallet, do the following:

1. Copy the target gateway self signed wallet, for example, `cwallet.sso`, containing the certificates for the Oracle Database Gateway to Object Store.

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

2. On Autonomous AI Database, create credentials to access the Object Store where you store the `cwallet.sso`.

See CREATE_CREDENTIAL Procedure for information about the username and password parameters for different object storage services.

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

For example:

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

See Create directory in Autonomous AI Database for information on creating directories.

4. Use DBMS_CLOUD.GET_OBJECT to upload the target gateway self signed wallet to the directory you created in the previous step, DBLINK_WALLET_DIR.

For example:

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

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

Note

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

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

For example:

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

Supplying the `credential_name` parameter is mandatory.

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

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

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

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name => 'SALESLINK',
    hostname => 'example.com',
    port => '1522',
    service_name => 'example_service_name',
    ssl_server_cert_dn => 'ssl_server_cert_dn',
    credential_name => 'DB_LINK_CRED',
    directory_name => 'DBLINK_WALLET_DIR',
    gateway_link => TRUE,
    private_target => TRUE,
    gateway_params => NULL
  );
END;
/
```

If `directory_name` is not NULL, a TCPS-based database link is created.

The `private_target` parameter must be set to TRUE when the target non-Oracle Database is on a private endpoint (that is, the database link accesses a hostname that needs to be resolved in a VCN DNS server). When `private_target` is TRUE, the hostname parameter must be a single hostname (on a private endpoint, using an IP address, a SCAN IP, or a SCAN hostname is not supported).

When `gateway_link` is `TRUE` and `gateway_params` is `NULL`, this specifies that the database link is to a customer-managed Oracle gateway.

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

See `CREATE_DATABASE_LINK` Procedure for more details.

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

For example:

```
SELECT * FROM employees@SALES LINK;
```

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

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

Where `password` is the new password.

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

See `UPDATE_CREDENTIAL` Procedure for more details.

Querying External Data with Autonomous AI Database on Dedicated Exadata Infrastructure

Describes packages and tools to query and validate data with Autonomous AI Database on Dedicated Exadata Infrastructure.

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.

① 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.

Query External Data

To query data in files in the Cloud, you need to first store your object storage credentials in your Autonomous AI Database, 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:

```
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 => 'adb_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 delimited 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:

```
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 `reject limit` 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](#).

External Table Metadata Columns

The external table metadata helps you determine where data is coming from when you perform a query.

The external tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`, `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE`, or `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from.

- `file$path`: Specifies the file path text up to the beginning of the object name.
- `file$name`: Specifies the object name, including all the text that follows the final `"/`.

For example:

```
SELECT genre_id, name, file$name, file$path FROM ext_genre
WHERE rownum <= 2;
```

genre_id	name	file\$name	file\$path
1	Action	genre.csv	https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_gold/o/genre
2	Adventure	genre.csv	https://objectstorage.us-ashburn-1.oraclecloud.com/n/namespace-string/b/moviestream_gold/o/genre

See [Invisible Columns](#) for more information on invisible columns.

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:

```
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:

```
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_NAME	OWNER_NAME	TYPE	STATUS	START_TIME	UPDATE_TIME
LOGFILE_TABLE	BADFILE_TABLE				
-----	-----	-----	-----	-----	-----
CHANNELS_EXT	SH	VALIDATE	COMPLETED	13-NOV-17...	13-NOV-17...
VALIDATE\$21_LOG	VALIDATE\$21_BAD				

Using this SQL statement with the `WHERE` clause on the `TYPE` column displays all of the load operations with type `VALIDATE`.

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

```

SELECT * FROM VALIDATE$21_LOG;

```

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

```

SELECT * FROM VALIDATE$21_BAD;

```

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

Note

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

Query External Data on Oracle Compute Cloud at Customer (C3)

Describes packages and tools to allow the Autonomous AI Database deployments on Exadata Cloud@Customer to load, query, and manage unstructured data by building external tables and external partitioned tables from objects in Data Lakes.

You can use the DBMS_CLOUD package to configure Autonomous AI Database deployments on Exadata Cloud@Customer to access the unstructured data stored on the Oracle Compute Cloud at Customer (C3) Object Storage Bucket.

Overview

You require a data platform that can extract business insights from structured and unstructured data. The combination of structured data available in a data warehouse and unstructured data available in a Data Lake is known as Data Lakehouse.

An Autonomous Data Lakehouse allows data analytics on both structured and unstructured data. The structured data is stored in an Autonomous AI Lakehouse, while the unstructured data is stored in a Data Lake configured on C3 Object Storage Buckets. The Autonomous AI Lakehouse can query, load, and manage objects in the Data Lake by using a database package called DBMS_CLOUD. This package allows the Autonomous AI Database to create external and external partitioned tables on data lake objects such as csv, txt, Avro, and Parquet files. You can build Autonomous Data Lakehouses by combining the capabilities of Autonomous AI Database deployments on Exadata Cloud@Customer and Oracle C3.

An Autonomous Data Lakehouse has two main components: An Autonomous AI Lakehouse and a Data Lake. The Autonomous AI Lakehouse is built using Autonomous AI Database deployments on Exadata Cloud@Customer, while the Data Lake is built using C3 Object Storage Buckets. The Autonomous AI Database hosts the structured relational data, while the Data Lake hosts collections of unstructured data in the form of txt, csv, Avro, Parquet, and other file types. You can query the Autonomous AI Database, and the database determines where to find the results, whether in the structured data, the unstructured data, or a combination of the two. You can use the DBMS_CLOUD package to configure Autonomous AI Database deployments on Exadata Cloud@Customer to load, query, and manage unstructured data by building external and external partitioned tables from objects in Data Lakes.

Prerequisites

Follow the steps below to configure C3 Object Storage Buckets and to provision various Autonomous AI Database components. You need to finish these steps before configuring Autonomous AI Database to communicate with C3 Object Storage Buckets.

- Create a [C3 Object Storage Bucket](#).
- Create a user and group. Add the user to the group and configure the required policies to allow the user to manage files (add/read/remove) to the C3 Object Storage Bucket.
Example:

```
Allow group <group_name> to manage objects in tenancy where  
target.bucket.name='<new_bucket_name_created>'
```

- [Create an API-Key](#) for the user created. From the API-key generation process, note the following information: user_ocid, tenancy_ocid, fingerprint, region. In addition, save the Public and Private PEM key files created during the API-key generation process. This information is required to configure the DBMS_CLOUD package later.
- Download the https certificate authority chain as shown below. This certificate authority chain allows client applications to authenticate the C3 Object Storage Bucket via https protocol. The https certificates are usually saved on a crt file, for example c3_ca.crt. Obtain the certificate authority chain by pointing a browser to the following URL: https://iaas.domainname/cachain where domainname is the name of the DNS domain configured during the C3 infrastructure deployment.
- Create a new [Compute VM](#). Once the new VM starts, copy the Private PEM key file and the C3 https certificate file created in the above steps to the VM.
- Create an Exadata Infrastructure Resource.
- Create an Autonomous VM Cluster.
- Create an Autonomous Container Database (ACD).
- Create an Autonomous AI Database.

Configuring Autonomous AI Database to communicate with the C3 Object Storage Bucket

You need to complete the following two tasks for the Autonomous AI Database to communicate with the C3 Object Storage Bucket:

- Add the C3 Object Storage https certificate to the Autonomous VM Cluster Grid Infrastructure TCPS Wallet. For example, the wallet location is /var/opt/oracle/dbaas_acfs/grid/tcps_wallets. This is needed so that the Autonomous AI Database can authenticate the https connection to the C3 Object Storage.
- Add an entry to the C##CLOUD\$SERVICE.dbms_cloud_store table on the ACD created above. This is to tell the DBMS_CLOUD package that the C3 Object Storage bucket is a valid cloud store. A sample SQL command to do this is:

```
SQL>INSERT INTO C##CLOUD$SERVICE.dbms_cloud_store  
VALUES('ORACLE_BMC','<objectstorage.mydomain.com>',null,1);
```

In Autonomous AI Databases, you cannot complete the above two tasks as a regular user. Oracle Cloud Operations needs to run these two tasks on behalf of the regular user. You need to log in to My Oracle Support (MOS) to create a new Support Request for Oracle Cloud Operations to perform the two tasks. You need to add the following information to the ticket. You will get these values after you complete the steps in .

- OCID of Exadata Infrastructure resource
- OCID of the Autonomous VM Cluster
- OCID of the ACD
- C3 Object Storage URL
- C3 Object Storage https certificate file

After Cloud Operations completes the above tasks, log in to the Autonomous AI Database. Use the DBMS_CLOUD.CREATE_CREDENTIAL procedure to provide the Autonomous AI Database with the authentication information required to connect to the C3 Object Storage Bucket. See CREATE_CREDENTIAL for more details.

Query External Data with Data Catalog

Oracle Cloud Infrastructure Data Catalog is the metadata management service for Oracle Cloud that helps you discover data and support data governance. It provides an inventory of assets, a business glossary, and a common metastore for data lakes.

Autonomous AI Database can leverage this metadata to dramatically simplify management for access to your data lake's object store. Instead of manually defining external tables to access your data lake, use the external tables that are defined and managed automatically. These tables will be found in Autonomous AI Database-protected schemas that are kept up to date with changes in Data Catalog.

About Querying with Data Catalog

By synchronizing with Data Catalog metadata, Autonomous AI Database automatically creates external tables for each logical entity harvested by Data Catalog. These external tables are defined in database schemas that are fully managed by the metadata synchronization process. Users can immediately query data without having to manually derive the schema (columns and data types) for external data sources and manually create external tables.

Synchronization is dynamic, keeping the Autonomous AI Database up-to-date with respect to changes to the underlying data, reducing administration cost as it automatically maintains hundreds to thousands of tables. It also allows multiple Autonomous AI Database instances to share the same Data Catalog, further reducing management costs and providing a common set of business definitions.

The Data Catalog folders/buckets are containers that sync with Autonomous Database schemas. Logical entities within those folders/buckets map to Autonomous Database external tables. These schemas and external tables are automatically generated and maintained through the sync process:

- Folders/Buckets map to database schemas that are for organizational purposes only.
- The organization is meant to be consistent with the data lake and minimize confusion when accessing data thru different paths.
- Data Catalog is the source of truth for the tables contained within schemas. Changes made in the Data Catalog update the schema's tables during a subsequent sync.

To use this capability, a Database Data Catalog Administrator initiates a connection to a Data Catalog instance, selects which data assets and logical entities to synchronize, and runs the sync. The sync process creates schemas and external tables based on the selected Data Catalog harvested data assets and logical entities. As soon as the external tables are created, Data Analysts can start querying their data without having to manually derive the schema for external data sources and create external tables.

① Note

The DBMS_DCAT Package is available for performing the tasks required to query Data Catalog object store data assets. See DBMS_DCAT package.

Concepts Related to Querying with Data Catalog

An understanding of the following concepts is necessary for querying with Data Catalog.

Data Catalog

Data Catalog harvests data assets that point to the object store data sources you want to query with Autonomous AI Database. From Data Catalog you can specify how the data is organized during harvesting, supporting different file organization patterns. As part of the Data Catalog harvesting process, you can select the buckets and files you want to manage within the asset. For further information, see [Data Catalog Overview](#).

Object Stores

Object Stores have buckets containing a variety of objects. Some common types of objects found in these buckets include: CSV, parquet, avro, json, and ORC files. Buckets generally have a structure or a design pattern to the objects they contain. There are many different ways to structure data and many different ways of interpreting these patterns.

For example, a typical design pattern uses top-level folders that represent tables. Files within a given folder share the same schema and contain data for that table. Subfolders are often used to represent table partitions (for example, a subfolder for each day). Data Catalog refers to each top-level folder as a logical entity, and this logical entity maps to an Autonomous AI Database external table.

Connection

A connection is an Autonomous AI Database connection to a Data Catalog instance. For each Autonomous AI Database instance there can be connections to multiple Data Catalog instances. The Autonomous AI Database credential must have rights to access Data Catalog assets that have been harvested from object storage.

Harvest

A Data Catalog process that scans object storage and generates the logical entities from your data sets.

Data Asset

A data asset in Data Catalog represents a data source, which includes databases, Oracle Object Storage, Kafka, and more. Autonomous AI Database leverages Oracle Object Storage assets for metadata synchronization.

Data Entity

A data entity in Data Catalog is a collection of data such as a database table or view, or a single file and normally has many attributes that describe its data.

Logical Entity

In Data Lakes, numerous files typically comprise a single logical entity. For example, you may have daily clickstream files, and these files share the same schema and file type.

A Data Catalog logical entity is a group of Object Storage files that are derived during harvesting by applying filename patterns that have been created and assigned to a data asset.

Data Object

A data object in Data Catalog refers to data assets and data entities.

Filename Pattern

In a data lake, data may be organized in different ways. Typically, folders capture files of the same schema and type. You must register to Data Catalog how your data is organized.

Filename patterns are used to identify how your data is organized. In Data Catalog, you can define filename patterns using regular expressions. When Data Catalog harvests a data asset with an assigned filename pattern, logical entities are created based on the filename pattern. By defining and assigning these patterns to data assets, multiple files can be grouped as logical entities based on the filename pattern.

Synchronize (Sync)

Autonomous AI Database performs synchronizations with Data Catalog to automatically keep its database up-to-date with respect to changes to the underlying data. Synchronization can be performed manually, or on a schedule.

The sync process creates schemas and external tables based on the Data Catalog data assets and logical entities. These schemas are protected, which means their metadata is managed by Data Catalog. If you want to alter the metadata, you must make the changes in Data Catalog. The Autonomous AI Database schemas will reflect any changes after the next sync is run. For further details, see [Synchronization Mapping](#).

Synchronization Mapping

The synchronization process creates and updates Autonomous AI Database schemas and external tables based on Data Catalog data assets, folders, logical entities, attributes and relevant custom overrides.

Data Catalog	Autonomous AI Database	Mapping Description
Data asset and folder (object storage bucket)	Schema name	Default values: By default, the generated schema name in Autonomous AI Database has the following format: DCAT\${dcat-con-id}<_>data-asset-name<_>folder-name< - dcat-con-id is the unique Data Catalog connection identifier. For details on specifying this identifier, see the dcat_con_id parameter in the DBMS_DCAT RUN_SYNC Procedure. - data-asset-name is the name of the Data Catalog data asset's name. - folder-name is the Data Catalog folder name. This folder maps to an object storage bucket. Customizations: The default data-asset-name and folder-name can be customized by defining custom properties, business names and display names to override these default names. - data-asset-name can be overridden by defining the oracle-db-schema-prefix custom property for the data asset in Data Catalog . - folder-name can be overridden by defining the oracle-db-schema custom property for the folder in Data Catalog , a business name or display name. The following attributes are used in order of precedence for generating the folder-name: - oracle-db-schema custom property - Business Name - Display Name Examples: - If the connection id is DataModels, the asset name is ObjectStorage, the folder name is HR and there are no custom property overrides, the derived schema name is: DCAT\$DATAMODELS_OBJECTSTORAGE_HR - If the data asset name is MYASSET, the folder name is MYFOLDER, and there are no custom property overrides, the schema name is: DCAT\$MYASSET_MYFOLDER - If the data asset has oracle-db-prefix = FIRSTASSET, and the folder has oracle-db-schema = FIRSTFOLDER, then the schema name is: DCAT\$FIRSTASSET_FIRSTFOLDER

Data Catalog	Autonomous AI Database	Mapping Description
Logical entity	External table	Logical entities are mapped to external tables. If the logical entity has a partitioned attribute, it is mapped to a partitioned external table. The external table name is derived from the corresponding logical entity's Display Name, or Business Name. If <code>oracle-db-schema</code> is set, then its value overrides all the names and custom properties of the corresponding folders and data assets. For example, if <code>oracle-db-schema</code> for an entity is set to <code>EntitySchema</code>, then the table is created in schema <code>DCAT\$ENTITYSCHEMA</code>.
Logical entity's attributes	External table columns	Column names: The external table column names are derived from the corresponding logical entity's attribute display names, or business names. For logical entities derived from Parquet, Avro, and ORC files, the column name is always the display name of the attribute as it represents the field name derived from the source files. For attributes corresponding to a logical entity derived from CSV files, the following attribute fields are used in order of precedence for generating the column name: 1. <code>oracle-db-column-name</code> 2. Business Name 3. Display Name Column type: The <code>oracle-db-column-type</code> custom property overrides the default column type that was derived by Data Catalog . For attributes corresponding to a logical entity derived from Avro files with <code>TIME_MICROS</code>, <code>TIME_MILLIS</code>, <code>TIMESTAMP_MICROS</code> or <code>TIMESTAMP_MILLIS</code> data types, you must set the <code>oracle-db-column-type</code> of the corresponding attribute in Data Catalog . Column length: The <code>oracle-db-column-length</code> custom property overrides the default column length for a string field that was derived by Data Catalog . Column precision: The <code>oracle-db-column-precision</code> custom property overrides the default precision for a number that was derived by Data Catalog . For attributes corresponding to a logical entity derived from Avro files with <code>TIME_MICROS</code>, <code>TIME_MILLIS</code>, <code>TIMESTAMP_MICROS</code> or <code>TIMESTAMP_MILLIS</code> data types, you must set the <code>oracle-db-column-precision</code> of the corresponding attribute in Data Catalog . Column scale: The <code>oracle-db-column-scale</code> custom property overrides the default scale for a number that was derived by Data Catalog .

Typical Workflow with Data Catalog

There is a typical workflow of actions performed by users who want to query with Data Catalog.

The Database Data Catalog Admin creates a connection between the Autonomous AI Database instance and a Data Catalog instance, then configures and runs a synchronization (sync) between the Data Catalog and Autonomous AI Database. The sync creates external tables and schemas in the Autonomous AI Database instance based on the synced Data Catalog contents.

The Database Data Catalog Query Admin or Database Admin grants READ access to the generated external tables so that Data Analysts and other database users can browse and query the external tables.

The table below describes each action in detail. For a description of the different user types included in this table, see Data Catalog Users and Roles.

Note

The DBMS_DCAT Package is available for performing the tasks required to query Data Catalog object store data assets. See DBMS_DCAT Package.

Action	Who is the user	Description
Create policies	Database Data Catalog Administrator	The Autonomous AI Database user credential must have the appropriate permissions to manage Data Catalog and to read from object storage. More information: Required Credentials and IAM Policies.
Create credentials	Database Data Catalog Administrator	Ensure database credentials are in place to access Data Catalog and to query object store. The user calls DBMS_CLOUD.CREATE_CREDENTIAL to create user credentials. More information: DBMS_CLOUD CREATE_CREDENTIAL Procedure, .
Create connections to Data Catalog	Database Data Catalog Administrator	To initiate a connection between an Autonomous AI Database instance and a Data Catalog instance the user calls DBMS_DCAT.SET_DATA_CATALOG_CONN to specify the target Data Catalog instance. Connections from an Autonomous AI Database instance to multiple Data Catalog instances is supported. The connection to the Data Catalog instance must use a user credential with sufficient Oracle Cloud Infrastructure (OCI) privileges. Once the connection has been made, the Data Catalog instance is updated with the DBMS_DCAT namespace and custom properties (if they do not already exist). The user can run a query to see the new connection including all current connections: <pre>select * from all_dcat_connections;</pre> More information: SET_DATA_CATALOG_CONN Procedure, UNSET_DATA_CATALOG_CONN Procedure.

Action	Who is the user	Description
Create a selective sync	Database Data Catalog Administrator	Create a sync job by selecting the Data Catalog objects to sync. The user can: • Select data assets/folders to sync. • Select individual logical entities to sync. • Preview the resulting external tables before syncing. • Change external tables (for example, the name) by modifying custom properties on Data Catalog. More information: See CREATE_SYNC_JOB Procedure , DROP_SYNC_JOB Procedure, Synchronization Mapping
Sync with Data Catalog	Database Data Catalog Administrator	The user initiates a sync operation. The sync is initiated manually through the DBMS_DCAT . RUN_SYNC procedure call, or automatically as part of a scheduled sync job. The sync operation creates, modifies and drops external tables and schemas according to the Data Catalog contents and sync selections. Manual configuration is applied using Data Catalog Custom Properties. More information: See DBMS_DCAT RUN_SYNC Procedure, CREATE_SYNC_JOB Procedure , Synchronization Mapping
Monitor sync and view logs	Database Data Catalog Administrator	The user can view the sync status by querying the USER_LOAD_OPERATIONS view. After the sync process has completed, the user can view a log of the sync results, including details about the mappings of logical entities to external tables.
Grant privileges	Database Data Catalog Query Administrator, Database Administrator	The database Data Catalog Query Administrator or database Administrator must grant READ on generated external tables to data analyst users. This allows the data analysts to query the generated external tables.
Browse and query external tables	Data Analyst	Data analysts are able to query the external tables through any tool or application that supports Oracle SQL. Data Analysts can review the synced schemas and tables in the DCAT\$* schemas, and query the tables using Oracle SQL. More information: Synchronization Mapping
Terminate connections to Data Catalog	Database Data Catalog Administrator	To remove an existing Data Catalog association, the user calls the UNSET_DATA_CATALOG_CONN procedure. This action is only done when you no longer plan on using Data Catalog and the external tables that are derived from the catalog. This action deletes Data Catalog metadata, and drops synced external tables from the Autonomous AI Database instance. The custom properties on Data Catalog and OCI policies are not affected. More information: UNSET_DATA_CATALOG_CONN Procedure

Example: MovieStream Scenario

In this scenario, Moviestream is capturing data in a landing zone on object storage. Much of this data, but not necessarily all, is then used to feed an Autonomous AI Database. Prior to

feeding Autonomous AI Database, the data is transformed, cleansed and subsequently stored in the "gold" area.

Data Catalog is used to harvest these sources and then provide a business context to the data. Data Catalog metadata is shared with Autonomous AI Database, allowing Autonomous AI Database users to query those data sources using Oracle SQL. This data may be loaded into Autonomous AI Database or queried dynamically using external tables.

For more information on using Data Catalog, see [Data Catalog Documentation](#).

1. Object Store - Review buckets, folders and files

a. Review the buckets in your object store.

For example, below are the landing (moviestream_landing) and gold zone (moviestream_gold) buckets in object storage:

b. Review the folders and files in the object store buckets.

For example, below are the folders in the landing bucket (moviestream_landing) in object storage:

2. Data Catalog - Create filename patterns

- a. Inform Data Catalog how your data is organized using filename patterns. These are regular expressions used to categorize files. The filename patterns are used by the Data Catalog harvester to derive logical entities. The following two filename patterns are used to harvest the buckets in the MovieStream example. See [Harvesting Object Storage Files as Logical Data Entities](#) for further details on creating filename patterns.

Hive-style	Folder-style
<code>{bucketName:.*/}{logicalEntity:[^/]+}.db/{logicalEntity:[^/]+}/.*</code>	<code>{bucketName:[\w]+}/{logicalEntity:[^/]+}(?!.db)/.*\$</code>
- Creates logical entities for sources that contain ".db" as the first part of the object name. - To ensure uniqueness within the bucket, the resulting name is (db-name).(folder name)	- Creates a logical entity based on the folder name off of the root - To prevent duplication with Hive, object names that have ".db" in them are skipped.

- b. To create filename patterns, go to the **Filename Patterns** tab for your Data Catalog and click **Create Filename Pattern**. For example, the following is the **Create Filename Pattern** tab for the moviestream Data Catalog:

3. Data Catalog - Data Asset Creation

- a. Create a data asset that is used to harvest data from your object store.

For example, a data asset named phoenixObjStore is created in the moviestream Data Catalog:

- b. Add a connection to your data asset.

In this example, the data asset connects to the compartment for the moviestream object storage resource.

- c. Now, associate your filename patterns with your data asset. Select **Assign Filename Patterns**, check the patterns you want and click **Assign**.

For example, here are the patterns assigned to the phoenixObjStore data asset:

[Help](#)

Assign Filename Patterns

 When you unassign a filename pattern, the logical data entities already harvested using this filename pattern are marked inactive.

Select the filename patterns you want to associate with the phoenixObjStore data asset. Deselect to unassign a filename pattern already associated with the data asset. When you harvest the data asset, the selected filename patterns will be used to group multiple files into logical data entities based on filename patterns.

	Name	Description	Expression
☒	Hive	Pattern used to captur...	{bucketName:.*}/{logic...
☐	Single CSV File	A single CSV file repr...	{bucketName: [S]+}/{lo...
☒	Folder-based Logical E...	Each folder off the ro...	{bucketName: [w]+}/{lo...

Page 1 of 1 (1-3 of 3 items)
 <
>
1
<
>

Assign
Cancel

4. Data Catalog - Harvest data from object store

- a. Harvest the Data Catalog data asset. Select the object store buckets containing the source data.

In this example, the moviestream_gold and moviestream_landing buckets from object store are selected for harvesting.

- b. After running the job, you see the logical entities. Use the **Browse Data Assets** to review them.

In this example, you are looking at the customer-extension logical entity and its attributes.

If you have a glossary, Data Catalog recommends categories and terms to associate with the entity and its attributes. This provides a business context for the items. Schemas, tables and columns are oftentimes not self-explanatory.

In our example, we want to differentiate between the different types of buckets and the meaning of their content:

- what is a landing zone?
- how accurate is the data?

- when was it last updated?
- what is the definition of a logical entity or its attribute

5. Autonomous AI Database - Connect to Data Catalog

Connect Autonomous AI Database to Data Catalog. You need to ensure that the credential used to make that connection is using an OCI principal that is authorized to access the Data Catalog asset. For further information, see [Data Catalog Policies](#).

a. Connect to Data Catalog

```
BEGIN
DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'OCI_NATIVE_CRED',
    user_ocid       =>
'ocid1.user.oc1..aaaaaaatfn77fe3fxux3o5lego7glqjejrzsqsrs64f4jsjrhbsk5
qzndq',
    tenancy_ocid    =>
'ocid1.tenancy.oc1..aaaaaaaapwkfqz3upqklvmelbm3j77nn3y7uqmlsod75rea5zmtm
bl574ve6a',
    private_key     =>
'MIIEogIBAAKCAQEA...t9SH7Zx7a5iV7QZJS5WeFLMUEv+YbYAJnXK+dOnPQtkhoblQWCEY
3Hsblj7Xz7o=',
    fingerprint     =>
'4f:0c:d6:b7:f2:43:3c:08:df:62:e3:b2:27:2e:3c:7a');
END;
-- Variables are used to simplify usage later
define oci_credential = 'OCI_NATIVE_CRED'
define dcat_ocid = 'ocid1.datacatalog.oc1.iad.aaaaaaaardp66bg...twiq'
define dcat_region='us-ashburn-1'
define uri_root = 'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/mytenancy/b/landing/o'
define uri_private = 'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/mytenancy/b/private_data/o'

-- Query a private bucket to test the privileges.
select *
from dbms_cloud.list_objects('&oci_credential', '&uri_private/');

-----
-- Set the credentials to use for object store and data catalog
-- Connect to Data Catalog
-- Review connection
-----
-- Set credentials
exec dbms_dcat.set_data_catalog_credential(credential_name =>
'&oci_credential');
exec dbms_dcat.set_object_store_credential(credential_name =>
'&oci_credential');

-- Connect to Data Catalog
begin
    dbms_dcat.set_data_catalog_conn (
        region => '&dcat_region',
        catalog_id => '&dcat_ocid');
end;
/
```

```
-- Review the connection
select * from all_dcat_connections;
```

- b. Sync Data Catalog with Autonomous AI Database. Here, we'll sync all the object storage assets:

```
-- Sync Data Catalog with Autonomous Database
---- Let's sync all of the assets.
begin
  dbms_dcat.run_sync('{"asset_list":["*"]}');
end;
/

-- View log
select type, start_time, status, logfile_table from
user_load_operations; -- Logfile_Table will have the name of the table
containing the full log.
select * from dbms_dcat$_log;

-- View the new external tables
select * from dcat_entities;
select * from dcat_attributes;
```

- c. Autonomous AI Database - Now start running queries against object store.

```
-- Query the Data !
select *from dcat$phoenixobjstore_moviestream_gold.genre
;
```

6. Change schemas for objects

The default schema names are rather complicated. Let's simplify them by specifying both the asset and the folder's Oracle-Db-Schema custom attribute in Data Catalog. Change the data asset to PHX and the folders to landing and gold respectively. The schema is a concatenation of the two.

- a. From Data Catalog, navigate to the moviestream_landing bucket and change the asset to landing and gold respectively.

Before change:

After change:

- b. Run another sync.

Example: Partitioned Data Scenario

This scenario illustrates how to create external tables in Autonomous AI Database that are based on Data Catalog logical entities harvested from partitioned data in Object Store.

The following example is based on [Example: MovieStream Scenario](#) and has been adapted to demonstrate integrating with partitioned data. Data Catalog is used to harvest these sources and then provide a business context to the data. For further details about this example, see [Example: MovieStream Scenario](#).

For more information on using Data Catalog, see [Data Catalog Documentation](#).

1. Object Store - Review buckets, folders and files

- a. Review the buckets in your object store.

For example, below are the landing (moviestream_landing) and gold zone (moviestream_gold) buckets in object storage:

- b. Review the folders and files in the object store buckets.

For example, below are the folders in the landing bucket (moviestream_landing) in object storage:

2. Data Catalog - Create filename patterns

- a. Inform Data Catalog how your data is organized using filename patterns. These are folder prefixes or regular expressions used to categorize files. The filename patterns are used by the Data Catalog harvester to derive logical entities. When a folder prefix is specified, the Data Catalog automatically generates logical entities from the specified folder prefix in the object store. The following filename pattern is used to harvest the buckets in the MovieStream example. See [Harvesting Object Storage Files as Logical Data Entities](#) for further details on creating filename patterns.

Folder prefix	Description
workshop.db/	Creates logical entities for sources that contain "workshop.db" path in the object store.

- b. To create filename patterns, go to the **Filename Patterns** tab for your Data Catalog and click **Create Filename Pattern**. For example, the following is the **Create Filename Pattern** tab for the moviestream Data Catalog:

3. Data Catalog - Data Asset Creation

- a. Create a data asset that is used to harvest data from your object store.

For example, a data asset named amsterdamObjStore is created in the moviestream Data Catalog:

- b. Add a connection to your data asset.

In this example, the data asset connects to the compartment for the moviestream object storage resource.

- c. Now, associate your filename patterns with your data asset. Select **Assign Filename Patterns**, check the patterns you want and click **Assign**.

For example, here are the patterns assigned to the amsterdamObjStore data asset:

4. Data Catalog - Harvest data from object store

- a. Harvest the Data Catalog data asset. Select the object store buckets containing the source data.

In this example, the moviestream_gold and moviestream_landing buckets from object store are selected for harvesting.

- b. After running the job, you see the logical entities. Use the **Browse Data Assets** to review them.

In this example, you are looking at the sales_sample_parquet logical entity and its attributes. Note that Data Catalog has identified the month attribute as partitioned.

5. Autonomous AI Database - Connect to Data Catalog

Connect Autonomous AI Database to Data Catalog. You need to ensure that the credential used to make that connection is using an OCI principal that is authorized to access the Data Catalog asset. For further information, see [Data Catalog Policies](#).

a. Connect to Data Catalog

```

BEGIN
DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'OCI_NATIVE_CRED',    user_ocid          =>
'ocid1.user.oc1..aaaaaaaatfn77fe3fxux3o5lego7glqjejrzsqsrs64f4jsjrhbsk5
qzndq',
    tenancy_ocid          =>
'ocid1.tenancy.oc1..aaaaaaaapwfkqz3upqklvmelbm3j77nn3y7uqmlsod75rea5zmtm
bl574ve6a',
    private_key           =>
'MIIEogIBAAKCAQEA...t9SH7Zx7a5iV7QZJS5WeFLMUEv+YbYAjnXK+d0nPQtkh0blQWCEY
3Hsblj7Xz7o=',
    fingerprint           =>
'4f:0c:d6:b7:f2:43:3c:08:df:62:e3:b2:27:2e:3c:7a');
END;
-- Variables are used to simplify usage later
define oci_credential = 'OCI_NATIVE_CRED'
define dcat_ocid = 'ocid1.datacatalog.oc1.eu-
amsterdam-1...leguurn3dmqa'
define dcat_region='eu-amsterdam-1'
define uri_root = 'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/mytenancy/b/landing/o'
define uri_private = 'https://objectstorage.us-
ashburn-1.oraclecloud.com/n/mytenancy/b/private_data/o'

-- Query a private bucket to test the privileges.
select *
from dbms_cloud.list_objects('&oci_credential', '&uri_private/');

-----
-- Set the credentials to use for object store and data catalog
-- Connect to Data Catalog
-- Review connection
-----
-- Set credentials
exec dbms_dcat.set_data_catalog_credential(credential_name =>
'&oci_credential');
exec dbms_dcat.set_object_store_credential(credential_name =>
'&oci_credential');

-- Connect to Data Catalog
begin
    dbms_dcat.set_data_catalog_conn (
        region => '&dcat_region',
        catalog_id => '&dcat_ocid');
end;
/
-- Review the connection
select * from all_dcat_connections;

```

b. Sync Data Catalog with Autonomous AI Database. Here, we'll sync all the object storage assets:

```

-- Sync Data Catalog with Autonomous Database
---- Let's sync all of the assets.

```

```

begin
  dbms_dcat.run_sync('{ "asset_list":["*"]}');
end;
/

-- View log
select type, start_time, status, logfile_table from
user_load_operations; -- Logfile_Table will have the name of the table
containing the full log.
select * from dbms_dcat$log;

-- View the new external tables
select * from dcat_entities;
select * from dcat_attributes;

```

- c. Autonomous AI Database - Now start running queries against object store.

```

-- Query the Data !
select count(*) from
DCAT$AMSTERDAMOBJSTORE_MOVIESTREAM_LANDING.SALES_SAMPLE_PARQUET;

-- Examine the generated partitioned table
select
dbms_metadata.get_ddl('TABLE','SALES_SAMPLE_PARQUET','DCAT$AMSTERDAMOBJSTORE_MOVIESTREAM_LANDING') from dual;

CREATE TABLE
"DCAT$AMSTERDAMOBJSTORE_MOVIESTREAM_LANDING"."SALES_SAMPLE_PARQUET"
(
  "MONTH" VARCHAR2(4000) COLLATE "USING-NLS_COMP",
  "DAY_ID" TIMESTAMP (6),
  "GENRE_ID" NUMBER(20,0),
  "MOVIE_ID" NUMBER(20,0),
  "CUST_ID" NUMBER(20,0),
  ...
) DEFAULT COLLATION "USING-NLS_COMP"
ORGANIZATION EXTERNAL
( TYPE ORACLE_BIGDATA
  ACCESS PARAMETERS
    ( com.oracle.bigdata.fileformat=parquet
      com.oracle.bigdata.filename.columns=["MONTH"]
      com.oracle.bigdata.file_uri_list="https://swiftobjectstorage.eu-
amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/
sales_sample_parquet/*"
    ...
  )
)
REJECT LIMIT 0
PARTITION BY LIST ("MONTH")
(PARTITION "P1" VALUES (('2019-01'))
  LOCATION ( 'https://swiftobjectstorage.eu-
amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/
sales_sample_parquet/month=2019-01/*'),
PARTITION "P2" VALUES (('2019-02'))
  LOCATION ( 'https://swiftobjectstorage.eu-
amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/

```

```
sales_sample_parquet/month=2019-02/*'),
...PARTITION "P24" VALUES (('2020-12'))
  LOCATION ( 'https://swiftobjectstorage.eu-
amsterdam-1.oraclecloud.com/v1/tenancy/moviestream_landing/workshop.db/
sales_sample_parquet/month=2020-12/*'))
PARALLEL
```

6. Change schemas for objects

The default schema names are rather complicated. Let's simplify them by specifying both the asset and the folder's Oracle-Db-Schema custom attribute in Data Catalog. Change the data asset to PHX and the folders to landing and gold respectively. The schema is a concatenation of the two.

- a. From Data Catalog, navigate to the moviestream_landing bucket and change the asset to landing and gold respectively.

Before change:

After change:

- b. Run another sync.

Query External Data with AWS Glue Data Catalog

Autonomous AI Database supports a system for synchronizing with an Amazon AWS Glue Data Catalog instance.

About Querying with AWS Glue Data Catalog

Autonomous AI Database allows you to synchronize with Amazon Web Service (AWS) Glue Data Catalog metadata. A database external table is automatically created by Autonomous AI Database for every table harvested by AWS Glue about data stored in Amazon Simple Storage Service (S3). Users can query data stored in S3 from Autonomous AI Database without having to manually derive the schema for the external data sources and create external tables.

Amazon AWS Glue Data Catalog is a centralized metadata management service that helps data professionals discover data and supports data governance in AWS cloud. An Autonomous AI Database instance has the ability to synchronize automatically data catalog metadata with AWS Glue Data Catalog allowing database users to immediately use Autonomous AI Database to query data stored in the AWS cloud.

Synchronizing with AWS Glue Data Catalog has the same properties as synchronizing with OCI Data Catalog. Synchronization is dynamic, keeping the database up-to-date with respect to changes to the underlying data, reducing administration cost as it automatically maintains hundreds to thousands of tables.

Concepts Related to Querying with AWS Glue Data Catalog

An understanding of the following concepts is necessary for querying with Amazon Web Service (AWS) Glue data catalogs.

AWS Glue Data Catalog: Database

An AWS Glue database represents a collection of relational table definitions, organized in a logical group. Each AWS Glue data catalog instance manages multiple databases.

AWS Glue Data Catalog: Table

An AWS Glue table represents a relational table over data stored in the AWS cloud. An AWS Glue table defines the schema of the underlying data and consists of column information, partition information, serialization information, storage information, statistics, user-defined metadata and other metadata. Tables in AWS Glue data catalog can be created manually, or automatically using an AWS Glue crawler.

Glue Data Catalog: Crawler

You can use a crawler to populate the AWS Glue data catalog with tables. This is the primary method used by most AWS Glue users. A crawler can crawl multiple data stores in a single run. Upon completion, the crawler creates or updates one or more tables in your data catalog. Extract, transform, and load (ETL) jobs that you define in AWS Glue use these data catalog tables as sources and targets. The ETL job reads from and writes to the data stores that are specified in the source and target data catalog tables.

AWS Glue tables can be created manually by the user or automatically using a predefined or a custom crawler. Crawlers connect to the underlying data stores (for example, Amazon S3), invoke classifiers for deriving the schema of the data, and create AWS Glue tables for storing the inferred metadata. AWS Glue provides classifiers for common file types, such as CSV, JSON, Parquet, and AVRO.

Mapping Between Autonomous AI Database and AWS Glue

During the synchronization process, external tables are created in Autonomous AI Database derived from the AWS Glue Data Catalog databases and tables over Amazon S3.

AWS Glue organizes collected metadata in databases and tables. An AWS Glue database is a collection of relational table definitions. AWS Glue tables that describe the common schema and properties of the files associated with the table.

AWS Glue follows the relational model for representing attributes. For [mapping hierarchical schemas to relational schemas](#), AWS Glue infers the schema of the semi-structured data and flattens the data to a relational schema using an ETL process.

The following table represents the mapping between OCI Data Catalog concepts and AWS Glue Data Catalog Concepts.

Table 4-2 OCI Data Catalog to AWS Glue Data Catalog Mapping

OCI Data Catalog	AWS Glue Data Catalog	Oracle Database
Data Asset	Database	Schema
Folder	(Bucket)	Schema
Logical Entity	Table	Table

User Workflow for Querying with AWS Glue Data Catalog

The basic user workflow for querying AWS S3 data with AWS Glue Data Catalog involves connecting to AWS Glue Data Catalog, synchronizing with Autonomous AI Database to automatically create external tables, and then querying the S3 data.

The Database Data Catalog Administrator creates a connection between the Autonomous AI Database instance and an AWS Glue Data Catalog instance, then configures and runs a synchronization (sync) between the AWS Glue Data Catalog and Autonomous AI Database. Autonomous AI Database automatically creates an external table for tables harvested by AWS Glue about data stored in S3.

The Database Data Catalog Query Admin or Database Admin grants READ access to the generated external tables so that Data Analysts and other database users can browse and

query Autonomous AI Database without having to manually derive the schema for the external data sources and create external tables.

Users

The table below describes the different types of users that perform the user workflow actions.

User	Description
Database Data Catalog Administrator	Database user with DCAT_SYNC role.
Database Data Catalog Query Administrator	Database user able to grant access on automatically created external tables to other users.
Data Analyst	Database user on Autonomous AI Database querying data in AWS S3 either by querying automatically created external tables or by interacting directly with AWS Glue Data Catalog.
AWS Glue Data Catalog User	AWS user with access to an AWS Glue Data Catalog.
AWS S3 Object Storage User	AWS user with access to data stored in AWS S3

User Workflow

Note

The DBMS_DCAT package is available for performing the tasks required to query AWS S3 object storage using AWS Glue Data Catalog. See DBMS_DCAT Package.

Action	Who is the user	Description
Create policies	Database Data Catalog Administrator	The Autonomous AI Database user credential must have the appropriate permissions to access AWS Glue Data Catalog and to read from S3 object storage. More information: Required Credentials and IAM Policies.
Create credentials	Database Data Catalog Administrator	Ensure database credentials are in place to access AWS Glue Data Catalog and to query S3 object storage. The user calls DBMS_CLOUD.CREATE_CREDENTIAL to create user credentials.

Note

Only Amazon Web Services (AWS) credentials are supported. AWS Amazon Resource Names (ARN) credentials are not supported.

More information: DBMS_CLOUD CREATE_CREDENTIAL Procedure,

Action	Who is the user	Description
Connect	Database Data Catalog Administrator	Establish a connection between an Autonomous AI Database instance and an AWS Glue Data Catalog instance. The connection uses the privileges of the AWS Glue Data Catalog User. Connections from an Autonomous AI Database instance to multiple AWS Glue Data Catalog instances is supported. To initiate a connection between an Autonomous AI Database instance and an AWS Glue Data Catalog instance the user: 1. Calls <code>DBMS_DCAT.SET_DATA_CATALOG_CREDENTIAL</code> using an AWS credential (<code>DBMS_CLOUD.CREATE_CREDENTIAL</code> Procedure,) to specify the target AWS Glue Data Catalog instance. 2. Calls the <code>DBMS_DCAT.SET_OBJECT_STORE_CREDENTIAL</code> using an AWS credential (<code>DBMS_CLOUD.CREATE_CREDENTIAL</code> Procedure,) to access the AWS S3 object storage. 3. Calls <code>DBMS_DCAT.SET_DATA_CATALOG_CONN</code> using an AWS Glue service endpoint Once the connection has been made, Autonomous AI Database stores the associated metadata, such as the AWS Glue catalog id, region, endpoint, and credential objects. More information: <code>SET_DATA_CATALOG_CONN</code> Procedure, <code>UNSET_DATA_CATALOG_CONN</code> Procedure, <code>DBMS_DCAT.SET_DATA_CATALOG_CREDENTIAL</code>, <code>DBMS_DCAT.SET_OBJECT_STORE_CREDENTIAL</code>.
Synchronize	Database Data Catalog Administrator	The user can manually start a synchronization with connected AWS Glue Data Catalogs using <code>DBMS_DCAT.RUN_SYNC</code> or create automatic synchronizations using <code>DBMS_DCAT.CREATE_SYNC_JOB</code>. Synchronization does the following: • Creates external tables in Autonomous AI Database that are derived from the connected AWS Glue Data Catalog. • Allows users to preview metadata for AWS Glue databases and tables. • Modifies or drops existing synchronized external tables if AWS Glue metadata has changed. More information: See <code>DBMS_DCAT.RUN_SYNC</code> Procedure, <code>CREATE_SYNC_JOB</code> Procedure , <code>DROP_SYNC_JOB</code> Procedure, Synchronization Mapping
Monitor Synchronization	Database Data Catalog Administrator	The user can view the sync status by querying the <code>USER_LOAD_OPERATIONS</code> view. After the sync process has completed, the user can view a log of the sync results, including details about the mappings to external tables.
Grant privileges	Database Data Catalog Query Administrator, Database Administrator	The database Data Catalog Query Administrator or Database Administrator must grant <code>READ</code> privileges on generated external tables to data analyst users. This allows the data analysts to query the generated external tables.

Action	Who is the user	Description
Query	Data Analyst	Data analysts are able to review the synced schemas and tables in the GLUE\$* schemas and query the external tables through any tool or application that supports Oracle SQL. Data in S3 is accessed using the privileges of the AWS S3 object storage user. More information: Example: Query with AWS Glue Data Catalog
Terminate connections	Database Data Catalog Administrator	To remove an existing Data Catalog association, the user calls the UNSET_DATA_CATALOG_CONN procedure. This action is only done when you no longer plan on using the connected AWS Glue Data Catalog and the external tables that are derived from the catalog. This action deletes AWS Glue Data Catalog metadata, and drops synced external tables from the Autonomous AI Database instance. More information: UNSET_DATA_CATALOG_CONN Procedure

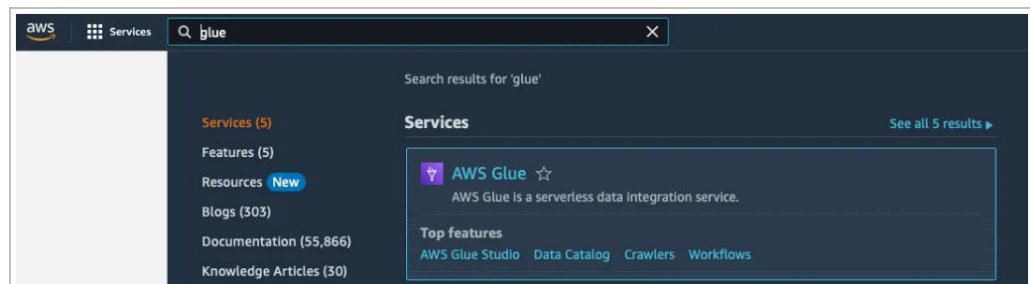
Example: Query with AWS Glue Data Catalog

This example steps you through the process of running queries over datasets stored in Amazon Simple Storage Service (Amazon S3) using an AWS Glue Data Catalog.

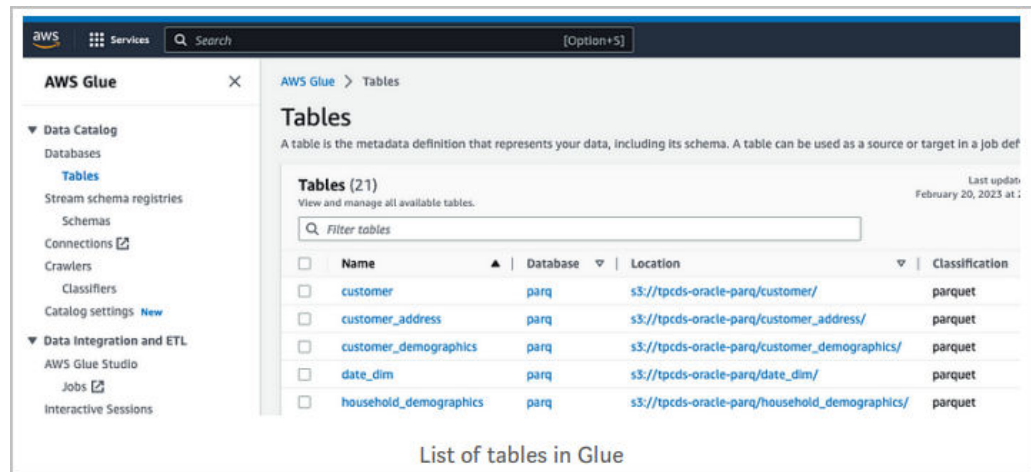
In this example, metadata in an AWS Glue Data Catalog is inspected to see what Amazon S3 objects have been previously crawled and exist in the data catalog. Autonomous AI Database is then associated with the AWS Glue Data Catalog and Amazon S3. The data catalog is synchronized with Autonomous AI Database to create external tables over the datasets stored in Amazon S3. The external tables are used to query the datasets in Amazon S3.

1. Inspect metadata in AWS Glue Data Catalog.

a. Launch the AWS Glue console.



b. Navigate to the data catalog, databases and tables to find existing objects. In this example, some objects exist in Amazon S3 that AWS Glue has previously crawled and created tables for as shown below:



2. Associate AWS Glue with Autonomous AI Database.

a. Create credentials in Autonomous AI Database.

The following procedure call includes the access ID and secret key to provide Autonomous AI Database with access to the underlying data in Amazon S3.

```
exec dbms_cloud.create_credential('CRED_AWS', '<access id>', '<access key>');
```

b. Associate the credentials with the AWS Glue Data Catalog and Amazon S3 object storage.

These procedure calls associate the data catalog and object storage, respectively, with the credentials.

```
exec dbms_dcat.set_data_catalog_credential('CRED_AWS');
exec dbms_dcat.set_object_store_credential('CRED_AWS');
```

c. Set up an AWS region where Glue is running.

```
exec dbms_dcat.set_data_catalog_conn(region => 'us-west-2',
catalog_type=>'AWS_GLUE');
```

3. Synchronize metadata to create external tables in Autonomous AI Database derived from AWS Glue databases and tables.

a. Now that the association is done, use the `all_glue_databases` view to find what databases are inside an AWS Glue Data Catalog.

```
select * from all_glue_databases order by name;
```

b. Use the `all_glue_tables` view to get a list of tables available for sync.

```
select * from all_glue_tables order by database_name, name;
```

CATALOG	DATA...	NAME	TABLE_TYPE	CLASSIFICATION	DESCRIPTION	OWNER	CREATED_BY
221152...	parq	customer	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	customer_address	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	customer_demographics	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	date_dim	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	household_demographics	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	income_band	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	inventory	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	item	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	promotion	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	reason	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	ship_mode	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	store	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	store_returns	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	store_sales	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	test_glue	EXTERNAL_TABLE	(null)	(null)	221152299178	arn:aws:iam::22115...
221152...	parq	time_dim	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...
221152...	parq	warehouse	EXTERNAL_TABLE	parquet	(null)	owner	arn:aws:sts::22115...

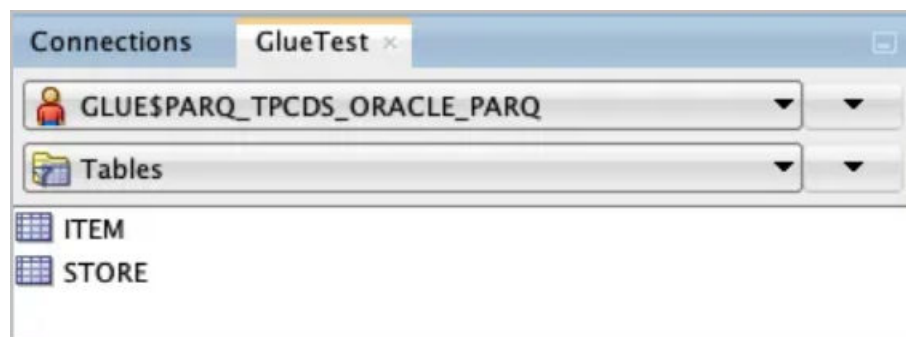
- c. Synchronize Autonomous AI Database with two tables, store and item, found in the parq database.

```

begin
  dbms_dcat.run_sync(
    synced_objects => '
      {
        "database_list": [
          {
            "database": "parq",
            "table_list": ["store","item"]
          }
        ]
      }',
    error_semantics => 'STOP_ON_ERROR');
end;
/

```

4. Inspect new objects in Autonomous AI Database and run a query on top of S3.
- Use SQL Developer to view new objects created by the previous sync operation. The GLUE\$PARQ_TPCDS_ORACLE_PARQ schema was generated and named automatically by the dbms_dcat.run_sync procedure call.



- Run a SQL query over datasets stores in Amazon S3.

```
SELECT * FROM glue$parq_tpcds_oracle_parq.store;
```

About Querying Apache Iceberg Tables

Autonomous AI Database supports querying of Apache Iceberg tables.

Note

Support for querying Apache Iceberg tables is available in Oracle Database 19c starting with version 19.28, and in Oracle Database 23ai starting with version 23.9.

Supported Configurations

These specific configurations are supported:

- Iceberg tables on Amazon Web Services AWS:
 - Iceberg tables registered with AWS Glue Data Catalog, created with Spark or Athena. For more information see [Use the AWS Glue connector to read and write Apache Iceberg tables with ACID transactions and perform time travel](#) and [Using Iceberg tables](#).
 - Iceberg tables stored on AWS S3 by directly providing the URL for the root metadata file.
- Iceberg tables on Oracle Cloud Infrastructure (OCI) Object Storage:
 - Iceberg tables generated with OCI Data Flow using a Hadoop Catalog. For more information, see [Oracle Data Flow Examples](#) and [Using a Hadoop Catalog](#).
 - Iceberg tables stored on OCI Object Storage by directly providing the URL for the root metadata file.

Restrictions

- Partitioned Iceberg tables
Oracle does not support Iceberg partitioned tables.
- Row-level updates of Iceberg tables
Oracle does not support merge-on-read for Iceberg table updates. Queries encountering deleted files in the Iceberg metadata will fail. For more information about merge-on-read, see [Enum RowLevelOperationMode](#).
- Schema evolution
The schema for Oracle external tables is fixed and reflects the Iceberg schema version at external table creation time. Queries fail if the Iceberg metadata points to a different schema version compared to the one used to create the Iceberg table. If the Iceberg schema changes after the external table has been created, it is recommended that the external table be recreated.

Concepts Related to Querying Apache Iceberg Tables

An understanding of the following concepts is helpful for querying Apache Iceberg tables.

Iceberg Catalog

The Iceberg catalog is a service that manages table metadata, such table snapshots, the table schema and partitioning information. In order to query the latest snapshot for an Iceberg table, query engines must first access the catalog and obtain the location of the most recent metadata file. There are already a number of available catalog implementations, including

AWS Glue, Hive, Nessie, and Hadoop. Autonomous AI Database supports the AWS Glue catalog and the HadoopCatalog implementation used by Spark.

For more information, see [Optimistic Concurrency](#).

Metadata Files

The metadata file is a JSON document that keeps track of the table snapshots, partitioning scheme and schema information. The metadata file is the entry point to a hierarchy of manifest lists and manifest files. The manifests track the table's data files along with information including partitioning and column statistics. See the [Iceberg Table Specification](#), for more information.

Transactions

Iceberg supports row-level updates to tables using either copy-on-write or merge-on-read. Copy-on-write generates new data files that reflect the updated rows, while merge-on-read generates new "delete files" that must be merged with the data files during reading. Oracle supports copy-on-write. Queries on iceberg tables fail if they encounter a delete file. For more information, see [RowLevelOperationMode](#).

Schema Evolution

Iceberg supports schema evolution. Schema changes are reflected in the Iceberg metadata using a schema ID. Note that Oracle external tables have a fixed schema, determined by the most current schema version at table creation time. Iceberg queries fail when the queried metadata points to a different schema version compared to the one used at table creation time. For more information, see [Schema Evolution](#).

Partitioning

Iceberg supports advanced partitioning options such as hidden partitioning and partition evolution that rely on processing/altering the table's metadata without costly data layout changes.

Examples: Querying Apache Iceberg Tables

These examples show how to query Apache Iceberg tables on Amazon Web Services (AWS) and Oracle Cloud Infrastructure (OCI), using a data catalog and using direct URLs for the root manifest file.

For detailed information on creating external tables for Apache Iceberg, see [#unique_547](#) CREATE_EXTERNAL_TABLE Procedure for Apache Iceberg.

Query an Iceberg table on AWS using a Glue Data Catalog

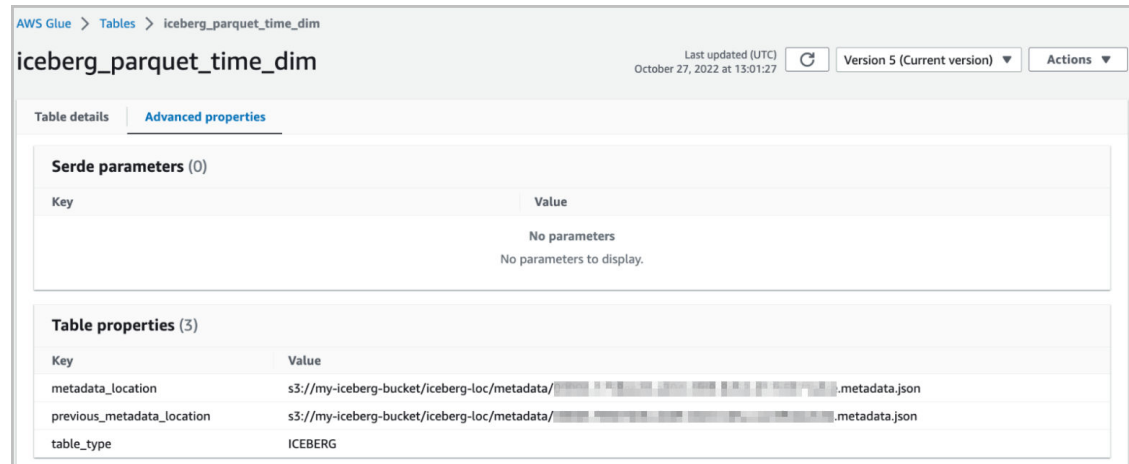
In this example, we query the Iceberg table `iceberg_parquet_time_dim`.

The screenshot shows the AWS Glue console interface for the table 'iceberg_parquet_time_dim'. The breadcrumb navigation is 'AWS Glue > Tables > iceberg_parquet_time_dim'. The table name is displayed prominently, along with 'Last updated (UTC) October 27, 2022 at 13:01:27', a refresh icon, 'Version 5 (Current version)', and an 'Actions' dropdown menu. Below this, there are two tabs: 'Table details' (selected) and 'Advanced properties'. The 'Table details' tab contains a table with the following information:

Name	Description	Database	Classification
iceberg_parquet_time_dim	-	my-iceberg-db	-
Location	Connection	Deprecated	Last updated
s3://my-iceberg-bucket/iceberg-loc	-	-	October 27, 2022 at 13:01:27
Input format	Output format	Serde serialization lib	
-	-	-	

The table belongs to Glue database my-iceberg-db and is stored in folder s3://my-iceberg-bucket/iceberg-loc.

The table details for iceberg_parquet_time_dim are shown here:



iceberg_parquet_time_dim	
Last updated (UTC) October 27, 2022 at 13:01:27	
Version 5 (Current version)	
Actions	
Table details Advanced properties	
Serde parameters (0)	
Key	Value
No parameters	
No parameters to display.	
Table properties (3)	
Key	Value
metadata_location	s3://my-iceberg-bucket/iceberg-loc/metadata/...metadata.json
previous_metadata_location	s3://my-iceberg-bucket/iceberg-loc/metadata/...metadata.json
table_type	ICEBERG

We can create an external table for iceberg_parquet_time_dim as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name      => 'iceberg_parquet_time_dim',
    credential_name => 'AWS_CRED',
    file_uri_list   => '',
    format          =>
        '{"access_protocol":
        {
            "protocol_type": "iceberg",
            "protocol_config":
            {
                "iceberg_catalog_type": "aws_glue",
                "iceberg_glue_region": "us-west-1",
                "iceberg_table_path": "my-iceberg-db.iceberg_parquet_time_dim"
            }
        }
        }',
    );
END;
/
```

In the protocol_config section we specify that the table uses AWS Glue as the catalog type, and set the catalog's region to us-west-1.

The credential AWS_CRED is created using dbms_cloud.create_credential with an AWS API key. The Glue Data Catalog instance is determined by the account ID associated with AWS_CRED for region us-west-1, as there is a single Glue Data Catalog region for each account. The iceberg_table_path element in the protocol_config section uses a \$database_name.\$table_name path to specify the Glue table name and database name. Finally, the column_list and field_list parameters are left null because the table's schema is automatically derived from the Iceberg metadata.

For further information on creating the credential, see CREATE_CREDENTIAL Procedure.

For information on AWS Glue resources, see [Specifying AWS Glue resource ARNs](#).

Query an Iceberg table on AWS using the location of the root metadata file

If we know the location of the metadata file for an Iceberg table, we can create an external table without specifying a catalog, as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name      => 'iceberg_parquet_time_dim',
    credential_name => 'AWS_CRED',
    file_uri_list   => 'https://my-iceberg-bucket.s3.us-west-1.amazonaws.com/
iceberg-loc/metadata/00004-1758ee2d-a204-4fd9-8d52-
d17e5371a5ce.metadata.json',
    format         => '{"access_protocol":{"protocol_type":"iceberg"}}');
END;
/
```

We use the `file_uri_list` parameter to specify the location of the metadata file in AWS S3 virtual-hosted-style URL format. For information on this format, see [Methods for accessing an AWS S3 bucket](#).

In this example, the database accesses the metadata file directly, so there is no need to provide a `protocol_config` section in the format parameter. When using the metadata file location to create an external table, the database queries the most recent snapshot referenced by the metadata file. Subsequent updates to the Iceberg table, that create new snapshots and new metadata files, will not be visible to the database.

Query an Iceberg table that uses Hadoop Catalog on OCI

In this example, we query the Iceberg table `icebergTablePy`, created using OCI Data Flow, where Spark uses the HadoopCatalog implementation for the Iceberg catalog. HadoopCatalog uses a warehouse directory and puts the Iceberg metadata in a `$database_name/$table_name` subfolder under this directory. It also uses a `version-hint.text` file that contains the version number for the most recent metadata file version. See [Iceberg Support on OCI Data Flow](#) for the example on Github.

The sample table `db.icebergTablePy` was created using a warehouse folder, named `iceberg`, in OCI bucket `my-iceberg-bucket`. The storage layout on OCI for table `icebergTablePy` is shown below:

Location: iceberg/db/icebergTablePy/metadata

Upload

More Actions

Search by prefix

☐	Name	Last Modified	Size	Storage Tier
☐	                                			

```

file_uri_list    => '',
format          => '{"access_protocol":{"protocol_type":"iceberg",
      "protocol_config":{"iceberg_catalog_type": "hadoop",
      "iceberg_warehouse":"https://objectstorage.uk-
cardiff-1.oraclecloud.com/n/my-tenancy/b/my-iceberg-bucket/o/iceberg",
      "iceberg_table_path": "db.icebergTablePy"}}}';
END;
/

```

Query an Iceberg table on OCI using the location of the root metadata file

We can query the Iceberg table described in the previous section by directly using the URL for the metadata file, as follows:

```

BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
  table_name      => 'iceberg_parquet_time_dim4',
  credential_name => 'OCI_CRED',
  file_uri_list   => 'https://objectstorage.uk-
cardiff-1.oraclecloud.com/n/my-tenancy/b/my-iceberg-bucket/o/iceberg/db/
icebergTablePy/metadata/v2.metadata.json',
  format          => '{"access_protocol":{"protocol_type":"iceberg"}}'
);
END;
/

```

In this example, we use the `file_uri_list` parameter to specify the URI for the metadata file using the native OCI URI format. When using the metadata file URI, the external table always queries the latest snapshot stored in the specific file. Subsequent updates that generate new snapshots and new metadata files are not accessible to the query.

For more information on the native OCI URI format, see [Cloud Object Storage URI Formats](#).

Integrate

Analyzing and Visualizing Data

Gain insight into the data in your Autonomous AI Database with Oracle Analytics Desktop and Oracle Analytics Cloud. These tools let you explore your data through advanced analytics and interactive visualizations.

Use Oracle Analytics Desktop to Analyze and Visualize Data

Oracle Analytics Desktop makes it easy to visualize your Autonomous AI Database data so you can focus on exploring interesting data patterns.

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.

Just connect Oracle Analytics Desktop to your Autonomous AI Database, select the elements that you're interested in, and let it find the best way to visualize it. Choose from a variety of visualizations to look at data in different ways.

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 a database with Oracle Analytics Desktop, see *User's Guide for Oracle Analytics Desktop*.

To download Oracle Analytics Desktop, go to the [Oracle Analytics Desktop Download](#) page.

Use Oracle Analytics Cloud to Analyze and Visualize Data

You can use Oracle Analytics Cloud with Autonomous AI Database. Oracle Analytics Cloud provides a complete set of tools for deriving and sharing data insights.

- **Data preparation:** Analysts can ingest, profile, and cleanse data using a variety of algorithms.
- **Data flow:** Analysts can prepare, transform and aggregate data, and then run machine-learning models at scale.
- **Data discovery:** Subject matter experts can easily collaborate with other business users, blending intelligent analysis at scale, machine learning, and statistical modeling.
- **Data visualization:** Analysts can visualize any data, on any device, on premises and in the cloud.
- **Data collaboration:** Large organizations and small teams can share data more simply, as you don't need to manage or consolidate multiple versions of spreadsheets.
- **Data-driven:** Application developers can utilize interfaces that enable them to extend, customize, and embed rich analytic experiences in the application flow.

To connect Oracle Analytics Cloud to your Autonomous AI Database follow *Lab 24: Oracle Analytics Cloud* in Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop to perform these high-level steps.:

1. Download, install and configure Oracle Analytics Cloud Data Gateway on a system that can connect to your database.
2. In Oracle Analytics Cloud, configure Remote Data Connectivity using information about your Data Gateway installation.
3. In Oracle Analytics Cloud, create a connection to your Autonomous AI Database

To learn about and use Oracle Analytics Cloud, visit the [Oracle Analytics Cloud](#) home page.

Oracle Extensions for IDEs

Oracle extensions let developers connect to, browse, and manage Autonomous AI Databases directly from common IDEs.

Use Oracle Cloud Infrastructure Toolkit for Eclipse

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

You can use the plugin to perform cloud and database operations right from Eclipse, such as creating Autonomous AI Databases, stopping and starting, scaling up and down, and so on. You can also use the plugin to easily connect to the databases to browse the schema, access tables, execute SQL statements, and perform other development tasks.

Users with permissions to manage the databases can perform a number of actions, including those listed below. For detailed information about permissions, see [Toolkit for Eclipse](#) in the Oracle Cloud Infrastructure documentation. You can:

- Create Autonomous AI Databases
- Start, stop, terminate, clone, and restore Autonomous AI Databases
- Scale up and down
- Download the client credentials zip file (database wallet)
- Connect to Autonomous AI Databases
- Browse the schema
- Choose compartments and regions
- Change the administrator password, update the license type, and so on

Download the latest version of the plugin from GitHub (`com.oracle.ocieclipse-version.zip`, where *version* is the latest version, for instance 1.2.0):

<https://github.com/oracle/oci-toolkit-eclipse/releases>

Then follow the installation instructions and details about how to get started in this step-by-step walkthrough:

[New Eclipse Plugin for Accessing Autonomous AI Database](#)

Use Oracle Developer Tools for Visual Studio

Oracle Developer Tools for Visual Studio is a tightly integrated extension for Microsoft Visual Studio and Oracle Autonomous AI Database on Dedicated Exadata Infrastructure. The extension is free and supports Visual Studio 2019 and Visual Studio 2017 on Microsoft Windows.

You can use the extension to perform database management operations right from Visual Studio, such as creating Autonomous AI Databases, stopping and starting, scaling up and down, and so on. You can also use the extension to easily connect to the databases and perform development tasks, such as browsing your Oracle schema and launching integrated Oracle designers and wizards to create and alter schema objects.

Users with permissions to manage the databases can perform a number of actions, including the following:

- Sign up for Oracle Cloud
- Connect to a cloud account using a simple auto-generated config file and key file
- Create new or clone existing Always Free Autonomous AI Database, Autonomous AI Database Dedicated, and Autonomous AI Database Serverless databases
- Automatically download credentials files (including wallets) and quickly connect, browse, and operate on Autonomous AI Database schemas
- Change compartments and regions without reconnecting
- Start, stop, or terminate Autonomous AI Database
- Scale up/down Autonomous AI Database resources
- Restore from backup
- Update instance credentials, update the license type used
- Rotate wallets
- Convert Always Free Autonomous AI Database into paid databases

Note

Promotion of Always Free to paid is currently supported *only* if the Always Free instance has database release 19c.

Download the extension from Visual Studio Marketplace:

- [Oracle Developer Tools for Visual Studio 2019](#)
- [Oracle Developer Tools for Visual Studio 2017](#)

You'll find lots of information about the extension on those Marketplace pages.

Then follow the installation instructions and details about how to get started in this step-by-step walkthrough:

[New Release: Visual Studio Integration with Oracle Autonomous AI Database](#)

For detailed information about how to use the extension, see the online documentation that's optionally installed with Oracle Developer Tools for Visual Studio. Press the F1 key to display the context-sensitive help for each dialog.

Use Oracle Developer Tools for VS Code

Oracle Developer Tools for VS Code is a tightly integrated extension for Microsoft Visual Studio Code (VS Code) and Oracle Autonomous AI Database on Dedicated Exadata Infrastructure. The extension is free and is available for Linux, Microsoft Windows, and Apple Mac OS.

You can use the extension to connect to Autonomous AI Databases right from Visual Studio Code and easily explore database schema, view table data, and edit and execute SQL and PL/SQL.

Download the extension from Visual Studio Marketplace:

[Oracle Developer Tools for VS Code](#)

Installation instructions and information about how to get started can be found in this quick start guide:

[Getting Started Using Oracle Developer Tools for VS Code](#)

Connect Power BI and Microsoft Data Tools to Autonomous AI Database on Dedicated Exadata Infrastructure

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

OCMT supports the following Microsoft tools:

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

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

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

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

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

- Provision an Autonomous AI Database. Refer to [Create an Autonomous AI Database for instructions](#).
- Download database credentials to the Windows client. Refer to [Download Client Credentials for instructions](#).
- Depending on the Microsoft (MS) tool that you want to connect to the Autonomous AI Database, you can install any of the following MS tools to the Windows client:
 - Visual Studio and Microsoft Reporting Services extension
 - Visual Studio and Microsoft Analysis Services extension
 - Visual Studio with the SQL Server Integration Services extension, which also supports SQL Server Data Tools

- Power BI On-premises Data gateway
- Power BI Desktop
- Install and configure ODP.NET on the Windows client
- Validate the MS tool connection to Autonomous AI Database

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

- [Power BI Desktop: Connect to Oracle Database](#)
- [Power BI Service: Connect to Oracle Database](#)
- [SQL Server Data Tools and Integration Services: Connect to Oracle Database](#)
- [SQL Server Analysis Services: Connect to Oracle Database](#)
- [SQL Server Reporting Services: Connect to Oracle Database](#)

Use Select AI for Natural Language Interaction with your Database

Use Select AI to Generate SQL from Natural Language Prompts

Autonomous AI Database on Dedicated Exadata Infrastructure Select AI enables you to query your data using natural language.

The Select AI feature allows Autonomous AI Database to use generative AI with Large Language Models (LLMs) to convert user's input text into Oracle SQL. Select AI processes the natural language prompt, supplements the prompt with metadata, and then generates and runs a SQL query.

About Select AI

Use natural language to interact with your database and LLMs through SQL to enhance user productivity and develop AI-based applications. Select AI simplifies and automates using generative AI, whether generating, running, and explaining SQL from a natural language prompt, using retrieval augmented generation with vector stores, generating synthetic data, or chatting with the LLM.

When you use Select AI, Autonomous AI Database manages the process of converting natural language into SQL. This means you can provide a natural language prompt instead of SQL code to interact with your data. Select AI serves as a productivity tool for SQL users and developers and enables non-expert SQL users to derive useful insights from their data, without having to understand data structures or technical languages.

Select AI also automates the retrieval augmented generation (RAG) process from generating vector embeddings to retrieving relevant content based on your prompt through semantic similarity search using your vector store. Other features include synthetic data generation, supporting chat history for conversations, and other features, all from a SQL interface.

The DBMS_CLOUD_AI package enables integration with a user-specified LLM for generating SQL code using natural language prompts. For natural language to SQL generation, this package provides an augmented prompt to the LLM containing the relevant database schema metadata. This enables generating, running, and explaining SQL queries based on natural language prompts. It also facilitates retrieval augmented generation using vector stores, synthetic data generation, and allows for chatting with the LLM. The DBMS_CLOUD_AI package works with AI providers listed in [Select your AI Provider and LLMs](#).

Note

- You must have an account with the AI provider and provide the credentials through DBMS_CLOUD_AI objects that the Autonomous AI Database uses.
- You can submit prompts in multiple languages. The quality of the result depends on the abilities of the specific LLM or the embedding model (transformer) being used. Check your LLM or embedding model documentation for multi-language support.

Supported Platforms

Select AI is supported on Autonomous AI Database Serverless and Autonomous AI Database on Dedicated Exadata Infrastructure and Cloud at Customers.

- Autonomous AI Database Serverless
- Autonomous AI Database on Dedicated Exadata Infrastructure
- Autonomous AI Database on Dedicated Exadata Infrastructure Region
- Autonomous AI Database Cloud@Customer

Related Terminology

It is important to understand the various terms used with Select AI before using it.

The following are the terms related to Select AI feature:

Term	Definition
Conversations	Conversations in Select AI represent an interactive exchange between the user and the system, enabling users to query or interact with the database through a series of natural language prompts. Select AI incorporates session-based short-term conversations to generate context-aware responses for the current prompt based on prior interactions. Up to 10 previous prompts are incorporated into the current request with short-term conversations, creating an augmented prompt that is sent to the LLM. Select AI supports using customizable long-term conversations enabling you to use Select AI with different topics without mixing context, that can be configured through conversation APIs from the DBMS_CLOUD_AI Package. See Use and Customize Conversations for more details.
Database Credential	Database Credentials are authentication credentials used to access and interact with databases. They typically consist of a user name and a password, sometimes supplemented by additional authentication factors like security tokens. These credentials are used to establish a secure connection between an application or user and a database, ensuring that only authorized individuals or systems can access and manipulate the data stored within the database.
Hallucination in LLM	Hallucination in the context of Large Language Models refers to a phenomenon where the model generates text that is incorrect, nonsensical, or unrelated to the input prompt. Despite being a result of the model's attempt to generate coherent text, these instances can contain information that is fabricated, misleading, or purely imaginative. Hallucination can occur due to biases in training data, lack of proper context understanding, or limitations in the model's training process.

Term	Definition
IAM	Oracle Cloud Infrastructure Identity and Access Management (IAM) lets you control who has access to your cloud resources. You can control what type of access a group of users have and to which specific resources. To learn more, see Overview of Identity and Access Management .
Iterative Refinement	Iterative refinement is a process of gradually improving a solution or a model through repeated cycles of adjustments based on feedback or evaluation. It starts with an initial approximation, refines it step by step, and continues until the desired accuracy or outcome is achieved. Each iteration builds on the previous one, incorporating corrections or optimizations to move closer to the goal. In text summary generation, iterative refinement can be useful for processing large files or documents. The process splits the text into manageable-sized chunks, for example, that fit within an LLM's token limits, generates a summary for one chunk, and then improves the summary by sequentially incorporating the following chunks. Use cases for iterative refinement: • Best suited for situations where contextual accuracy and coherence are critical, such as when summarizing complex or highly interconnected texts where each part builds on the previous. • Ideal for smaller-scale tasks where sequential processing is acceptable. See Summarization Techniques.
MapReduce	In general, the MapReduce programming model enables processing large-volume data by dividing tasks into two phases: Map and Reduce. • Map: Processes input data and transforms it into key-value pairs. • Reduce: Aggregates and summarizes the mapped data based on keys. MapReduce performs parallel processing of large data sets. In the case of Select AI Summarize, MapReduce partitions text into multiple chunks and processes them in parallel and independently, generating individual summaries for each chunk. These summaries are then combined to form a cohesive overall summary. Use cases for map reduce: • Best suited for large-scale, parallel tasks where speed and scalability are priorities, such as summarizing very large data sets or documents. • Ideal for situations where chunk independence is acceptable, and the summaries can be aggregated later. See Summarization Techniques.
Metadata	Metadata is data that describes data. In the case of Select AI, metadata is database metadata, which refers to the data that describes the structure, organization, and properties of the database tables and views. For database tables and views, metadata includes column names and types, constraints and keys, view definitions, relationships, lineage, quality and freshness indicators, security classifications, and access policies. Well-managed metadata enables discoverability, correct usage, performance tuning, and compliance. Select AI augments NL2SQL prompts with table metadata that include the table definition (table name, columns names and their data types), and optionally table and column comments, annotations, and constraints.

Term	Definition
Metadata Enrichment	The practice of augmenting database schemas with high-quality descriptions, comments, and annotations so an LLM can better understand the intent for tables and columns, clarify business meaning, and generate more accurate SQL. It turns bare table or column names into well-documented assets with clear intent, relationships, and constraints. Candidate information to include: • Table and column descriptions: purpose, business definitions, units, and allowed value ranges • Keys and relationships: primary/foreign keys, join paths • Data semantics: time granularity, slowly changing dimensions, deduplication rules • Constraints and quality: nullability, uniqueness, validation rules, data freshness • Synonyms and aliases: common business terms that map to technical names • Examples and patterns: sample values, common filters or aggregations See Overview of AI Enrichment to learn more about adding such metadata using Oracle SQL Developer for VS Code through Visual Studio Code.
Natural Language Prompts	Natural Language Prompts are human-readable instructions or requests provided to guide generative AI models, such as Large Language Models. Instead of using specific programming languages or commands, users can interact with these models by entering prompts in a more conversational or natural language form. The models then generate output based on the provided prompt.
Network Access Control List (ACL)	A Network Access Control List is a set of rules or permissions that define what network traffic is allowed to pass through a network device, such as a router, firewall, or gateway. ACLs are used to control and filter incoming and outgoing traffic based on various criteria such as IP addresses, port numbers, and protocols. They play a crucial role in network security by enabling administrators to manage and restrict network traffic to prevent unauthorized access, potential attacks, and data breaches.
Similarity Search	Similarity search identifies and retrieves data points that closely match a given query by comparing feature vectors in a vector store.
Vector Distances	Vector distances measure the similarity or dissimilarity between feature vectors by calculating the distance between them in a multidimensional space.
Vector Index	A vector index organizes and stores feature vectors to enable efficient similarity search and retrieval of related data.
Vector Store	A vector store includes systems that store, manage, and enable semantic similarity search involving vector embeddings. This includes standalone vector databases and Oracle Database 23ai AI Vector Search.

Select AI Use Cases

Select AI enhances data interaction and enables developers to build AI-driven applications directly from SQL, transforming natural language prompts to SQL queries and text responses, supporting chat interaction with LLMs, enhancing response accuracy with current data using RAG, and generating synthetic data.

Use cases include:

- Generate SQL from natural language prompts

Developer productivity: Select AI significantly enhances developer productivity by providing "starter" SQL queries quickly. Developers can input natural language prompts, and Select AI generates SQL based on your database schema tables and views. This reduces the time and effort needed to write complex queries from scratch, allowing developers to focus on refining and optimizing the generated queries for their specific needs.

Natural language queries for end-users: Select AI empowers end-users to interact with your application's underlying data tables and views using natural language queries. This functionality allows users without SQL expertise to ask questions and retrieve data directly, making data access more intuitive and user-friendly relative to the capabilities of LLM being used and the quality of the schema metadata available.

Other capabilities for SQL generation: The following highlighted capabilities are also supported for natural language to SQL generation:

- Specify schema or tables or views: Select AI enables you to specify an object list consisting of schema and optionally tables or views within that schema.
- Automatically detect relevant table metadata: Select AI automatically detects relevant tables and send metadata only for those specific tables, relevant to the query, in Oracle Database 26ai.
- Restrict table access: Select AI enables you to restrict table access by considering only the tables listed in the AI profile attributes for SQL generation.
- Specify case sensitivity for columns: Select AI enables user to specify case sensitivity such that the LLM produces case-insensitive responses from the database and LLM.

- Conversations

Enable chatbot-like features with Select AI, enabling users to have natural conversations for querying data and performing actions. These chats can keep track of context, giving follow-up answers that clarify or expand on original questions. This scenario boosts engagement and makes complicated queries easier through conversation.

- Agentic workflows with Select AI Agent

Use Select AI Agent to coordinate agents, tools (SQL, RAG, Websearch, Notifications), and tasks for multi-step scenarios such as data retrieval, and notification. See Build Autonomous Agents with Select AI Agent to learn more.

- Customized media generation

Select AI can be used to generate personalized media content such as emails tailored to individual customer details. For example, in your prompt you could instruct the LLM to create a friendly and upbeat email encouraging a customer to try a set of recommended products. These recommendations could be based on customer demographics or other specific information available in your database. This level of customization enhances customer engagement by delivering relevant and appealing content directly to the customer.

- Code generation

With the Select AI chat action, you can use Select AI to ask your specified LLM to generate code from natural language prompts. This feature supports various programming languages such as SQL, Python, R, and Java. Examples include:

- **Python Code:** "Write the Python code to compute a confusion matrix over a DataFrame with columns ACTUAL and PREDICTED."

- **SQL DDL:** "Write the DDL for a SQL table with columns name, age, income, and country."
- **SQL Query:** "Write the SQL query that will use the Oracle Machine Learning in-database model named CHURN_DT_MODEL to predict which customers will churn and with what probability."
- Retrieval Augmented Generation (RAG)
Use vector store content for semantic similarity search to enhance prompt accuracy and relevance in LLM responses.
- Synthetic data generation
Generate synthetic data using LLMs that conforms to your schema for solution testing, proofs of concept, and other uses. Synthetic data can support better testing of your applications in the absence of real data, leading to overall quality of your application.

Usage Guidelines

Provides usage guidelines that ensure effective and proper usage of natural language prompts for SQL generation to ensure an enhanced user experience.

Intended Use

This feature is intended for the generation and running of SQL queries resulting from user-provided natural language prompts. It automates what a user could do manually based on their schema metadata in combination with a large language model (LLM) of their choice.

While any prompt can be provided, including those that do not relate to the production of SQL query results, Select AI focuses on SQL query generation. Select AI enables submitting general requests with the chat action.

Prompt Augmentation Data

The database augments the user-specified prompt with database metadata to mitigate hallucinations from the LLM. The augmented prompt is then sent to the user-specified LLM to produce the query.

The database augments the prompt with schema metadata only. This metadata may include schema definitions, table and column comments, and content available from the data dictionary and catalog. For the purposes of SQL generation, the database does not provide table or view contents (actual row or column values) when augmenting the prompt.

The narrate action, however, does provide the result of the query, which may contain database data, to the user-specified LLM from which to generate natural language text describing the query results.

 Warning

Large language models (LLMs) have been trained on a broad set of text documentation and content, typically from the Internet. As a result, LLMs may have incorporated patterns from invalid or malicious content, including SQL injection. Thus, while LLMs are adept at generating useful and relevant content, they also can generate incorrect and false information including SQL queries that produce inaccurate results and/or compromise security of your data.

The queries generated on your behalf by the user-specified LLM provider will be run in your database. Your use of this feature is solely at your own risk, and, notwithstanding any other terms and conditions related to the services provided by Oracle, constitutes your acceptance of that risk and express exclusion of Oracle's responsibility or liability for any damages resulting from that use.

Select your AI Provider and LLMs

Choose an AI provider and LLM that meets your security standards and aligns with your specific needs, like text or code generation.

Different LLMs excel at various tasks based on their training data and intended purpose. Some models are excellent for text generation but may not perform well in code generation, while others are specifically optimized for coding tasks. Choose an LLM that best suits your needs.

AI Provider	LLMs	Embedding Model for RAG	Purpose
OCI Generative AI	- meta.llama-3.3-70b-instruct (default) - meta.llama-3.2-90b-vision-instruct - meta.llama-3.2-11b-vision-instruct - meta.llama-3.1-70b-instruct - meta.llama-3.1-405b-instruct - cohere.command-r-08-2024 - cohere.command-r-plus-08-2024 - cohere.command-r-16k (deprecated) - cohere.command-r-plus (deprecated) - xai.grok-3 - xai.grok-3-fast See: About the Chat Models in Generative AI The xAI Platform for OCI Generative AI	- cohere.embed-english-v3.0 (default) - cohere.embed-multilingual-v3.0 - cohere.embed-english-light-v3.0 - cohere.embed-multilingual-light-v3.0 See About the Embedding Models in Generative AI .	The OCI Generative AI Chat models are supported for all SELECT AI actions such as <code>runsql</code>, <code>showsql</code>, <code>explainsql</code>, <code>narrate</code>, and <code>chat</code>. The OCI Generate text models are supported only for SELECT AI chat action. To configure your profile attributes, see Profile Attributes.

AI Provider	LLMs	Embedding Model for RAG	Purpose
Azure OpenAI Service	- GPT-4o - GPT-4 - GPT-4 Turbo with Vision - GPT-3.5-Turbo	text-embedding-ada-002	Best suited for generating SQL from natural language prompts, chat action, and Select AI RAG.
OpenAI	- gpt-3.5-turbo (default) - gpt-4o - gpt-4o-mini - gpt-4 - gpt-4-0613 - gpt-4-32k - gpt-4-32k-0613 - gpt-3.5-turbo-0613 - gpt-3.5-turbo-16k - gpt-3.5-turbo-16k-0613	text-embedding-ada-002	Best suited for generating SQL from natural language prompts, chat action, and Select AI RAG.
OpenAI-Compatible	Models from OpenAI-compatible providers such as: - Fireworks AI - xAI - Others	Embedding models from OpenAI-compatible providers. For example, see Fireworks AI embedding models .	Supports a wide range of use cases.
Cohere	- command (default) - command-nightly (experimental) - command-r - command-r-plus - command-light - command-light-nightly (experimental) - custom models	embed-english-v2.0	Best suited for chat action.
Google	- gemini-1.5-flash (default) - gemini-1.5-pro - gemini-1.0-pro	text-embedding-004 (default)	Best suited for generating SQL from natural language prompts, chat action, and Select AI RAG.
Anthropic	- claude-3-5-sonnet-20240620 (default) - claude-3-opus-20240229 - claude-3-sonnet-20240229 - claude-3-haiku-20240307	NA	Best suited for generating SQL from natural language prompts, chat action, and Select AI RAG.

AI Provider	LLMs	Embedding Model for RAG	Purpose
Hugging Face	- Mixtral-8x7B-Instruct-v0.1 (default) - Meta-Llama-3-70B-Instruct - Qwen1.5-1.8B - other chat models	NA	Best suited for generating SQL from natural language prompts, chat action, and Select AI RAG.
AWS	Foundation models Custom models	- amazon.titan-embed-text-v1 - amazon.titan-embed-text-v2:0 - cohere.embed-english-v3	Supports a wide range of use cases.

① Note

- Specify OpenAI-compatible provider through provider_endpoint instead of provider parameter. See Profile Attributes.
- For models that accept images, use meta.llama-3.2-90b-vision-instruct. This model is specifically trained for vision and images. While it can be used for text and SQL generation, the model is best suited for images. To learn more, see [Chat in OCI Generative AI](#).

The meta.llama-3.2-11b-vision-instruct model provides robust multimodal capabilities.

- Embedding models are also known as transformer models.

Configure AI Profiles Using DBMS_CLOUD_AI

Autonomous AI Database uses AI profiles to facilitate and configure access to an LLM and to setup for generating, running, and explaining SQL based on natural language prompts. It also facilitates retrieval augmented generation using embedding models and vector indexes and allows for chatting with the LLM.

AI profiles include database objects that are the target for natural language queries. Metadata used from these targets can include database table names, column names, column data types, and comments. You create and configure AI profiles using the DBMS_CLOUD_AI.CREATE_PROFILE and DBMS_CLOUD_AI.SET_PROFILE procedures.

Requirements for DBMS_CLOUD_AI

The following are required to run DBMS_CLOUD_AI:

- Access to an Oracle Cloud Infrastructure cloud account and to an Autonomous AI Database instance.
- A paid API account for a supported AI provider, one of:
 - OpenAI:** To enable OpenAI to generate SQL from natural language prompts, obtain API keys from your OpenAI paid account.

You can find your secret API key in your [User settings](#).

- **Cohere:** To enable Cohere to generate SQL from natural language prompts, obtain API keys from your Cohere paid account.

Click **Dashboard**, and click **API Keys** on the left navigation. Copy the default API key or create another key. See [API-Keys](#) for more information.
- **Azure OpenAI Service:** To enable Azure OpenAI Service to generate SQL from natural language prompts, configure and provide access to the AI provider.

To use Azure OpenAI Service, perform the following steps:
 1. Obtain your secret API keys. You can find your API keys in the Resource Management section of your Azure portal. On your Azure OpenAI Service Resource page, click **Keys and Endpoint**. You can copy either KEY1 or KEY2.
 2. Create an Azure OpenAI Service resource and deploy a model: [Create and deploy an Azure OpenAI Service resource](#).

✓ **Tip**

 - * Note the resource name and deployment name as those parameters are used to provide network access permission and creating your Azure OpenAI Service profile using the `DBMS_CLOUD_AI.CREATE_PROFILE` procedure.
 - * To know more about rate limits for token per minute on a model, see [Azure OpenAI Service quotas and limits](#).

- 3. Allow access to Azure OpenAI Service:
 - * You can use your secret API key to allow access to Azure OpenAI Service. To know more, see the example in [Examples of Using Select AI](#).
- **OCI Generative AI:** See [How to Generate the API Signing Key](#).
- **Google:** To enable Google AI Studio to generate SQL and text responses to your natural language prompts, obtain API keys from your Google AI Studio paid account.
 1. Go to [Google AI Studio](#).
 2. Click Sign In to Google AI Studio.
 3. Click Get API key on the prompt screen.
 4. Select all the applicable options on the next page.
 5. Click Create API key.
 6. Click Create API key in new project.The screen displays the progress and generates an API key. Copy the key and save it.
- **Anthropic:** To enable Anthropic Developer Console to generate SQL and text responses to your natural language prompts, obtain API keys from your Anthropic Developer Console paid account.
 1. Go to [Anthropic Developer Console](#).
 2. Sign up for an account if you don't have one already.
 3. Once logged in, navigate to the API section or the dashboard.
 4. Look for an option to generate or view API keys.
 5. Click to create a new API key.

6. Copy the generated API key and save it.

The Claude API is a paid service. You'll need to add credits to your account before you can use the API key.

- **Hugging Face:** To enable Hugging Face as your AI provider to generate SQL and text responses to your natural language prompts, obtain API keys from your Hugging Face paid account.

1. Go to [Hugging Face](#).
2. Sign up for an account if you don't have one already.
3. Navigate to your account settings.
4. In the navigation menu locate the Access Tokens.
5. Click to create a new API key.
6. Copy the generated API key and save it.

- **AWS:** To enable AWS, obtain your API key and model ID.

Obtain your API key and use it to create credentials through `DBMS_CLOUD.CREATE_CREDENTIAL`.

Follow this process to obtain your API key and model name:

1. Sign up for an AWS account if you don't have one already.
2. Create your access keys and secret keys from [AWS Bedrock Console](#).
3. Copy the generated keys and save it.
4. Request access to their foundation models. See [Access Amazon Bedrock foundation models](#).
5. Obtain the model ID. You require the model ID in `DBMS_CLOUD_AI.CREATE_PROFILE` procedure. Model ID depends on the resources that you use. If you use:
 - * a base model, specify the model ID or its ARN(Amazon Resource Names). For a list of model IDs for base models, see [Amazon Bedrock base model IDs](#)
 - * an inference profile, specify the inference profile ID or its ARN. For a list of inference profile IDs, see [Supported Regions and models for cross-region inference](#).
 - * a provisioned model, specify the ARN of the Provisioned Throughput. For more information, see [Run inference using a Provisioned Throughput](#).
 - * a custom model, purchase Provisioned Throughput for it. Then, specify the ARN of the resulting provisioned model. For more information, see [Use a custom model in Amazon Bedrock](#).
 - * an Amazon Bedrock Marketplace model, specify the ID or ARN of the marketplace endpoint that you created, see [Amazon Bedrock Marketplace](#).

Note

Imported model is not supported with Bedrock Converse API.

To use AWS as your provider, see [Example: Select AI with AWS](#).

- Network ACL privileges to access your external AI provider.

Note

Network ACL is not applicable for OCI Generative AI.

- A credential that provides access to the AI provider.

To configure DBMS_CLOUD_AI:

1. Grant the EXECUTE privilege on the DBMS_CLOUD_AI package to the user who wants to use Select AI.

By default, only ADMIN user is granted the EXECUTE privilege. The ADMIN user can grant EXECUTE privilege to other users.

The following example grants the EXECUTE privilege to ADB_USER:

```
grant execute on DBMS_CLOUD_AI to ADB_USER;
```

The following example grants ADB_USER the privilege to use the *api.openai.com* endpoint.

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'api.openai.com',
    ace => xs$ace_type(privilege_list => xs$name_list('http'),
                      principal_name => 'ADB_USER',
                      principal_type => xs_acl.ptype_db)
  );
END;
/
```

2. Grant network ACL access to the user who wants to use Select AI and for the AI provider endpoint.

The ADMIN user can grant network ACL access. See *APPEND_HOST_ACE Procedure* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for more information.

3. Create a credential to enable access to your AI provider.
See CREATE_CREDENTIAL Procedure for more information.

Here is an example of how to create a credential to enable access to OpenAI.

```
EXEC DBMS_CLOUD.CREATE_CREDENTIAL('OPENAI_CRED', 'OPENAI',
'your_api_token');
```

APPEND_HOST_ACE Function Parameters

Parameter	Description
host	The host, which can be the name or the IP address of the host. You can use a wildcard to specify a domain or a IP subnet. The host or domain name is not case sensitive. For OpenAI, use <i>api.openai.com</i>. For Cohere, use <i>api.cohere.ai</i>. For Azure OpenAI Service, use <i><azure_resource_name>.openai.azure.com</i>. See Profile Attributes to know more about <i>azure_resource_name</i>. For Google, use <i>generativelanguage.googleapis.com</i>. For Anthropic, use <i>api.anthropic.com</i>. For Hugging Face, use <i>api-inference.huggingface.co</i>.
ace	The access control entries (ACE). The XS\$ACE_TYPE type is provided to construct each ACE entry for the ACL. For more details, see <i>Creating ACLs and ACEs</i> in Oracle Database 19c Real Application Security Administrator's and Developer's Guide or Oracle Database 26ai Real Application Security Administrator's and Developer's Guide.

DBMS_CLOUD.CREATE_CREDENTIAL Parameters

Parameter	Description
credential_name	The name of the credential to be stored. The <i>credential_name</i> parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
username	The username and password arguments together specify your AI provider credentials. The username is a user-specified user name.
password	The username and password arguments together specify your AI provider credentials. The password is your AI provider secret API key, and depends on the provider, that is, OpenAI, Cohere, Azure OpenAI Service, OCI, Google, Anthropic, or Hugging Face. See for details.

Create and Set an AI Profile

Describes the steps to create and enable an AI profile.

Use `DBMS_CLOUD_AI.CREATE_PROFILE` to create an AI profile. Next start `DBMS_CLOUD_AI.SET_PROFILE` to enable the AI profile so that you can use `SELECT AI` with a natural language prompt.

Note

You must run `DBMS_CLOUD_AI.SET_PROFILE` in each new database session (connection) before you use `SELECT AI`.

The following example with the OpenAI provider creates an AI profile called OPENAI and sets the OPENAI profile for the current user session.

```
-- Create AI profile
--
SQL> BEGIN
  DBMS_CLOUD_AI.create_profile(
    'OPENAI',
    '{"provider": "openai",
     "credential_name": "OPENAI_CRED",
     "object_list": [{"owner": "SH", "name": "customers"},
                     {"owner": "SH", "name": "sales"},
                     {"owner": "SH", "name": "products"},
                     {"owner": "SH", "name": "countries"}]
    }');
END;
/
```

PL/SQL procedure successfully completed.

```
--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.set_profile('OPENAI');
```

PL/SQL procedure successfully completed.

Use AI Keyword to Enter Prompts

Use AI as the keyword in a SELECT statement for interacting with the database using natural language prompts.

The AI keyword in a SELECT statement instructs the SQL execution engine to use the LLM identified in the active AI profile to process natural language and to generate SQL.

You can use the AI keyword in a query with Oracle clients such as SQL Developer, OML Notebooks, and third-party tools, to interact with database in natural language.

Note

You cannot run PL/SQL statements, DDL statements, or DML statements using the AI keyword.

Syntax

The syntax for running AI prompt is:

```
SELECT AI action natural_language_prompt
```

Parameters

The following are the parameters available for the *action* parameter:

Parameter	Description
<code>runsql</code>	Run the provided SQL command using a natural language prompt. This is the default action and it is optional to specify this parameter.
<code>showsql</code>	Displays the SQL statement for a natural language prompt.
<code>narrate</code>	The output of the prompt is explained in natural language. This option sends the SQL result to the AI provider to produce a natural language summary.
<code>chat</code>	Generates a response directly from the LLM based on the prompt. If conversation in the <code>DBMS_CLOUD_AI.CREATE_PROFILE</code> function is set to <code>true</code> , this option includes content from prior interactions or prompts, potentially including schema metadata.
<code>explainsql</code>	The SQL generated from the prompt is explained in natural language. This option sends the generated SQL to the AI provider to produce a natural language explanation.

Usage Notes

- Select AI is not supported in Database Actions or APEX Service. You can use only `DBMS_CLOUD_AI.GENERATE` function.
- The AI keyword is supported only in a SELECT statement.
- You cannot run PL/SQL statements, DDL statements, or DML statements using the AI keyword.
- The sequence is SELECT followed by AI. These keywords are not case-sensitive. After a `DBMS_CLOUD_AI.SET_PROFILE` is configured, the text after SELECT AI is a natural language prompt. If an AI profile is not set, SELECT AI reports the following error:

```
ORA-00923: FROM keyword not found where expected
00923. 00000 - "FROM keyword not found where expected"
```

- Special character usage rules apply according to Oracle guidelines. For example, use single quotes twice if you are using an apostrophe in a sentence.

```
select ai how many customers in SF don't own their own home
```

- LLMs are subject to *hallucinations* and results are not always correct:
 - It is possible that SELECT AI may not be able to run the generated SQL for a specific natural language prompt.
 - It is possible that SELECT AI may not be able to generate SQL for a specific natural language prompt.

In such a scenario, SELECT AI responds with information to assist you in generating valid SQL.

- Use the chat action, with SELECT AI chat, to learn more about SQL constructs. For better results with the chat action, use database views or tables with contextual column names or consider adding column comments explaining values stored in the columns.
- To access DBA or USER views, see `DBMS_CLOUD_AI` Views.

Use and Customize Conversations

Conversations in Select AI refer to the interactive dialogue between the user and the system, where a sequence of user-provided natural language prompts are stored and managed to support long-term memory for LLM interactions.

Select AI supports short-term, session-based conversations, which are enabled in the AI profile, as well as long-term, named conversations, which are enabled using specific procedures or functions and conversation IDs.

Note

Support for Select AI Conversations is available in Oracle Database 19c starting with version 19.30, and in Oracle Database 26ai starting with version 23.26.1.

Types of Conversations

Select AI supports session-based short-term conversations and customizable conversations.

Session-based Short-Term Conversations: Select AI includes session-based short-term conversations to generate context-aware responses for the current prompt based on prior interactions.

You can enable it by setting the `conversation` attribute to `true|false` in your AI profile. Unlike the multiple conversation feature, session-based conversations store prompts only for the duration of the session. Prompts are stored in a temporary table, automatically dropped when the session ends, and cannot be reused and switched among conversations.

Customizable Long-Term Conversations: Select AI supports creating and using customizable conversations enabling you to use Select AI with different topics without mixing context, improving both flexibility and efficiency. You can create, set, delete, and update conversations through the `DBMS_CLOUD_AI` conversation procedures and functions. When you enable conversations, Select AI retrieves prompt history and sends them to the LLM to generate a response for the current prompt. These responses are stored in a persistent table for future use.

Note

Select AI Conversations support the following actions: `runsql`, `showsql`, `explainsql`, `narrate`, and `chat`.

How to Use Customizable Conversations

You can use Select AI for customizable conversations in the following ways:

- Set the conversation ID in the session using the `DBMS_CLOUD_AI.SET_CONVERSATION_ID` procedure, and run `SELECT AI <ACTION> <PROMPT>`.
- Pass the `conversation_id` in the `params` argument of the `DBMS_CLOUD_AI.GENERATE` function.

Note

If you use both multiple conversations and the conversation: [true|false] setting, the system ignores the conversation setting.

In the context of SQL query generation (NL2SQL), conversations enable a more intuitive and accessible way to work with your data, making it easier for users to extract insights and perform tasks without needing deep technical knowledge of SQL. Conversations can also be used with chat and RAG capabilities of Select AI.

For example, conversations provide an intuitive way to analyze data:

- Ask a question such as "What are the total number of customers"
- Follow up with context aware questions like:
 - "break out count of customers by country"
 - "What age group is most common"
 - "keep the top 5 customers and their country by their purchases and include a rank in the result"

To enable conversations, see [Example: Enable Conversations in Select AI](#) for a complete example.

Session-Based Conversations vs Customizable Conversations

The following table compares the session-based conversation and the customizable conversations in Select AI:

Questions	Session-Based Short-Term Conversations	Customizable Long-Term Conversations
When do I use?	Best for quick, temporary chats where you want the model to remember recent questions and answers during a single session. It's useful when you don't need to save or reuse the conversation later.	Designed for longer or ongoing conversations that may span multiple sessions. It's useful when you want to track, review, or manage the conversation history, or when different parts of an application need to access the same conversation context over time.
How do I enable?	Set {"conversation": true or false} in your AI profile.	Use the DBMS_CLOUD_AI.SET_CONVERSATION_ID procedure or theDBMS_CLOUD_AI.GENERATE
How many conversations are permitted?	One.	You can create multiple conversations. If you explicitly specify conversation IDs, you can alternate between them to associate prompts with the appropriate conversations as needed.

Questions	Session-Based Short-Term Conversations	Customizable Long-Term Conversations
Where are prompts stored and for how long?	Prompts are stored in a temporary table and dropped when the session ends.	Prompts are stored in a permanent table. Prompts are retained in the database for the number of days specified by the <code>retention_days</code> parameter in the <code>DBMS_CLOUD_AI.CREATE_CONVERSATION</code> procedure. After the retention period, the conversation and its prompts are automatically deleted. You can also manually delete prompts using the <code>DBMS_CLOUD_AI.DELETE_CONVERSATION_PROMPT</code> procedure.
How many prompts are stored and how many prompts are sent to the LLM?	A maximum of 10 prompts are stored and sent to the LLM. You cannot customize this limit.	All prompts are stored. By default, the system sends the 10 most recent prompts to the LLM. You can customize this using the <code>conversation_length</code> parameter. See <code>CREATE_CONVERSATION</code> Attributes.
Can I delete individual prompts?	No, you cannot delete individual prompt manually.	You can delete specific individual prompts by using the prompt id as specified in the <code>USER_CLOUD_AI_CONVERSATION_PROMPTS</code> View and using the <code>DBMS_CLOUD_AI.DELETE_CONVERSATION_PROMPT</code> procedure.
Are AI profiles used for retrieval of conversations?	Yes, Select AI retrieves prompts and responses previously generated using the same AI profile.	No, Select AI tracks the AI profile used when storing prompts and responses but does not restrict their retrieval based on that profile. It sends all conversation histories to the LLM to guide response generation, regardless of the profile used to generate them.
Where can I check history of prompts?	Prompts are saved in a temporary table under <code>CLOUD_USER</code> but are not accessible for querying.	You can query and review conversations and prompts through the <code>DBMS_CLOUD_AI</code> conversation views. See <code>DBMS_CLOUD_AI</code> Views for details.

Provide Feedback to Improve LLMs

Select AI enables you to provide feedback to help improve your selected LLM's ability to generate more accurate SQL queries.

Note

This feature is available only on Oracle Database 23ai. You can use it alongside the existing Select AI actions: `runsql`, `showsql`, and `explainsql`. Ensure that your AI profile is configured for NL2SQL generation and not RAG.

You can provide feedback to improve the accuracy of the generated SQL through the feedback action or the `DBMS_CLOUD_AI.FEEDBACK` procedure. Select AI creates a default vector index named `<profile_name>_FEEDBACK_VECINDEX` with default attributes when you use the feedback feature for the first time. This index helps refine future generated SQL based on the feedback provided. See [Vector Index for FEEDBACK](#) for more information.

Tip

Use the `DBMS_CLOUD_AI.FEEDBACK` procedure when your specified LLM fails to generate the correct SQL query or does not return the expected result from one of the NL2SQL SELECT AI actions. You can also use the `DBMS_CLOUD_AI.FEEDBACK` procedure to add the correct SQL directly to the vector table. This helps guide future SQL generation by serving as a reference for similar prompts.

The following types of feedback are accepted:

- **Positive feedback:** You approve and confirm the accuracy of the generated SQL. The system stores the confirmed query for future reference.
- **Negative feedback:** If the results from your NL2SQL SELECT AI action fails to generate the correct SQL, you can provide the necessary SQL query improvements using SELECT AI feedback `<feedback>` or using the `DBMS_CLOUD_AI.FEEDBACK` procedure by identifying errors in the query or providing guidance as to what was expected in prose. The system refines the query using an LLM and stores the refined version for future Select AI query generation. The stored refined queries are then used as hints and sent to the LLM as part of the augmented prompt.

Benefits of Using Feedback

The feedback action and procedure in Select AI introduces a prompt tuning mechanism that enhances the accuracy of SQL query generation.

The owner of the AI profile can provide feedback for generated SQL queries and the Select AI system *learns* from the user interaction over time. This learning involves amassing a repository of prompts and feedback content where vector search is used to identify prompts similar to your current prompt. The top matching examples are provided as metadata to the LLM as part of the augmented prompt. The following are the benefits:

- Can improve accuracy of SQL queries.
- Enables the owner of AI profile to provide feedback by confirming the correct queries or suggesting corrections by providing the semantic intention or business definitions to produce a correct query.
- Uses historical feedback as hints for future query generation thereby adapting to your needs.

How to Provide Feedback

Select AI enables you to provide feedback either by using the feedback action or by calling the `DBMS_CLOUD_AI.FEEDBACK` procedure.

Caution

Do not use the feedback action in applications where multiple users share database sessions under a single database user that owns the AI profile. The AI profile owner should provide feedback only after confirming that the corrected query is appropriate for all users of that profile.

You can provide feedback by including the following:

- `SQL_TEXT`: Include the feedback in quotes for your current SQL query or you can get the `SQL_TEXT` for a particular query by querying the `V$_MAPPED_SQL` view.
- `SQL_ID`: You can get the `SQL_ID` by querying the `V$_MAPPED_SQL` view.
- Use the last generated SQL: Provide your feedback in natural language.

Note

To use last sql, be sure to set server output off in Oracle SQL*Plus or Oracle SQLcl. You must have `READ` privilege on `sys.v_$session` and `v_$mapped_sql` tables.

```
GRANT READ ON SYS.V_$MAPPED_SQL TO ADB_USER;  
GRANT READ ON SYS.V_$SESSION TO ADB_USER;
```

See `FEEDBACK` Procedure and [Example: Select AI Feedback](#) to learn more.

Query Graph Data with Property Graphs

Select AI generates Property Graph Query (PGQ) on Oracle Property Graphs using natural language. It enables users to query graph data through the `GRAPH_TABLE` operator with minimal SQL knowledge.

Select AI extends its natural language to SQL (NL2SQL) capability to graph structures enabling you to query SQL Property Graphs using natural language. Select AI applies the `GRAPH_TABLE` operator to interpret relationships and attributes in graph-structured data. It generates SQL or PGQ graph queries based on the data objects defined in the AI profile. When a property graph is included in the AI profile, Select AI uses generative AI to build a PGQ query that references the graph through the `GRAPH_TABLE` operator. The LLM automatically receives the graph object's metadata such as `CREATE PROPERTY GRAPH` statements to generate accurate queries. When a table, view, or relational object is specified, Select AI generates a SQL query. This capability simplifies pattern-matching queries on graph data stored in Oracle AI Database and reduces dependency on manually constructing SQL queries.

When you create an AI profile with one or more property graphs defined in the `object_list` attribute, the LLM defined in AI profile interprets prompts by using the context of the specified property graphs. Select AI builds an augmented prompt that includes:

- Instructions to construct PGQ queries.
- Metadata describing the provided property graphs (from their CREATE PROPERTY GRAPH statements).

This augmented prompt is sent to the LLM. Select AI runs the query and returns results. If a property graph is specified along with other object types such as tables, schema, or views in the AI profile, Select AI raises an error.

SQL vs PGQ

With Select AI, the type of query generated depends on the objects defined in the `object_list` attribute of your AI profile.

- SQL Query: uses relational data such as schema, tables, or views.
- PGQ Query: uses property graphs and applies the `GRAPH_TABLE` operator for pattern matching.

See SQL Property Graph and SQL GRAPH Queries for more details.

Benefits of Using Select AI on Property Graphs

Database users can query property graphs using Select AI to generate graph queries from natural language, reducing manual work and improving understanding of graph relationships.

Key benefits include:

- NL2SQL: Select AI's NL2SQL capability now extends to graph queries enabling users to write natural language prompts such as *"Find customers who bought a dress"*.
- SQL or PGQ: Depending on the data object, Select AI automatically generates SQL or PGQ query.
- Productivity: Reduces time and effort in building graph queries using the `GRAPH_TABLE` operator.
- Conversations: Retains conversation context and queries a property graph.

Limitations

Select AI for Property Graphs does not support the following capabilities:

- Synthetic data generation
- Mixing of property graphs with other object types in the AI profile
- Unsupported and intermittent queries. See [Example: Sample Prompts for Property Graphs](#) for more details.

How to Use Select AI on Property Graphs

Select AI enables you to explore graph data by using the `DBMS_CLOUD_AI.GENERATE` function or by using `Select AI <action> <prompt>`.

After you have defined property graphs in the `object_list` attribute of your AI profile, you can use:

- `SELECT AI <ACTION> <PROMPT>` in the SQL command line to generate an output.
- `DBMS_CLOUD_AI.GENERATE` function and supply your prompt within the function.

The following are the supported actions: `runsql`, `showsql`, `explainsql`, `narrate`, and `showprompt`. Select AI for Property Graph also supports session-based short-term and customizable long-term conversations.

See [Example: Select AI for Property Graphs](#) and [Example: Sample Prompts for Property Graphs](#) to learn more.

Generate a Summary with Select AI

Select AI enables you to generate a summary of your text, especially large texts, generally supporting up to 1 GB using AI providers. You can extract key insights from texts or large files as per your specific needs. This feature uses the LLM specified in your AI profile to generate a summary for a given text.

Select AI offers summarization of your content in the following ways:

- Use `summarize` as the Select AI *action*. See [Use AI Keywords to Enter Prompts and Example: Select AI Actions](#) to learn more.
- Use the `DBMS_CLOUD_AI.SUMMARIZE` function to specify customizations such as summary length, format, extraction level for a summary and so on with summarization parameters. See [Example: Select AI Summarize](#) to learn more.
- Use the `DBMS_CLOUD_AI.GENERATE` function with the `summarize` action. See [GENERATE Function](#) to learn more.

① Note

- Select AI *summarize* is applicable for Autonomous Database 19c and Autonomous AI Database 26ai.
- Select AI *summarize* supports content up to 1 GB.
- Supported file formats: Files must be encoded in *UTF-8*.
 - For Autonomous AI Database 26ai:
Select AI supports a broader range of file formats. For details, see [Oracle Text Supported Document Formats](#).
 - For Autonomous Database 19c:
Select AI supports only text files. The content must be in plain text and Select AI does not support *json*, *html*, *docx*, or *pdf*.

Summarization Techniques

You can summarize a large document by entering the full text as a prompt for the LLM. However, since LLMs have token limits, Select AI applies different techniques to summarize large documents.

Select AI uses iterative refinement and MapReduce as summarization techniques. See [Related Terminology](#) for more details.

Based a given LLM's maximum token size, Select AI can break large content into smaller, easier-to-handle chunks that fit within the LLM's limits. Select AI can then use either the iterative refinement or map reduce technique to generate the summary. Select AI uses MapReduce as the default setting. See [SUMMARIZE Function](#) and [SUMMARIZE Parameters](#) for more details.

Translate Text with Select AI

With Select AI, you can use generative AI from the OCI translation service to translate your text into the language of your choice.

You can use this feature alongside the existing Select AI RAG. You can combine `translate` with `DBMS_CLOUD_AI.GENERATE` or `narrate` to use generative AI for producing translated outputs in your preferred language.

See [TRANSLATE Function](#), [GENERATE Function](#), and [Example: Select AI Translate](#) to learn more.

Benefits of Using Translate

The `translate` feature in Select AI enables you to translate input text in different languages for the following benefits:

- `Translate` improves usability by converting text into your preferred language, enabling you to work with the database more effectively.
- `Translate` lets you convert documents in one language into a language that may be better suited for your selected embedding model, producing better vectors for use with RAG.
- Automatically translate output into your preferred language when using `DBMS_CLOUD_AI.GENERATE` or `NARRATE`.

How to Use Translate

Select AI enables you to translate your text input into your preferred language by using the `translate` action or by calling the `DBMS_CLOUD_AI.GENERATE` or the `DBMS_CLOUD_AI.TRANSLATE` functions.

To use the Select AI translation feature, you must have the appropriate IAM policy permissions to access Oracle Cloud Infrastructure Language services.

Grant the permission to use `ai-service-language-family` resource in your IAM policy. An example policy statement to grant permission to a user group in a specific compartment is:

```
allow group <your group name> to use ai-service-language-family in
compartment <your_compartment>
```

- If using Resource Principal credential, assign the permission to the Dynamic Group.
- If using Private Key credential, assign the permission to the User Group.

A Dynamic Group identifies resources such as databases or functions by matching their OCIDs or tags, while a User Group contains individual IAM users.

Use a dynamic group when the policy applies to OCI resources, and use a user group when the policy applies to human users. For detailed steps to create dynamic and user groups, see [Managing Dynamic Groups](#).

See [Language Policies](#) for more information.

You can translate text using the following:

- Use `translate` as a Select AI action. Use `select ai translate <text>`. See [Example: Select AI Actions](#).
- Provide `translate` as an action within the `DBMS_CLOUD_AI.GENERATE` function. See [GENERATE Function](#).

- Use the `DBMS_CLOUD_AI.TRANSLATE` function. See [TRANSLATE Function](#).

See also [Example: Select AI Translate](#) to learn more.

Select AI for Python

Select AI for Python integrates generative AI capabilities into Autonomous AI Database workflows. Select AI for Python provides a client library, `select_ai`, that enables you to use `DBMS_CLOUD_AI` capabilities in Autonomous AI Database from Python. Select AI for Python supports enhanced generative AI workflows, summarization, feedback mechanisms, consistent metadata management, and agentic AI capabilities. It also supports Python 3.14 and includes an updated HTML documentation site ([New](#))

What You Can Do

- Connect to the database using synchronous or asynchronous connections
- Create and manage AI profiles to enable using AI models from a wide range of AI providers
- Use natural language to query your database through AI-based SQL generation
- Describe query results in natural language
- Create and manage conversations with prompt history
- Create and update vector indexes easily for use with built-in and automated Retrieval Augmented Generation (RAG) workflows
- Generate synthetic data for testing and analysis using generative AI
- Summarize text or query results
- Record and manage model feedback

Supported Platforms

Select AI for Python is certified for Autonomous Database 19c and Autonomous AI Database 26ai. Select AI for Python may work on other platforms, however, it is not certified.

Click <https://github.com/oracle/python-select-ai/issues> to report issues.

Note

Select AI Agent for Python is not available on Autonomous AI Database on Dedicated Exadata Infrastructure.

Supported Functions for Select AI Profile (Synchronous and Asynchronous)

When you send prompts through a profile, you can choose among several functions defined for AI profile objects. Some are as follows:

- **`create()`**: Create the AI profile in the database or replace if necessary.
- **`delete()`**: Remove the profile.
- **`generate()`**: Use the profile to process a prompt as per the chosen action.
- **`generate_synthetic_data()`**: Create synthetic data based on the supplied attributes.
- **`get_attributes()`**: Return the current profile attributes.
- **`run_sql()`**: generate and run SQL (default).

- `show_sql()`: generate SQL without running it.
- `explain_sql()`: provide explanation for generated SQL.
- `narrate()`: describe query results in natural language.
- `chat()`: engage in a freeform conversation.
- `show_prompt()`: display the constructed prompt sent to the generative AI model.
- `summarize()`: Produce a summary for provided content.
- `add_positive_feedback()`, `add_negative_feedback()`, `delete_feedback()`: Manage user feedback associated with generated queries.

For a complete list of functions, see [Select AI for Python](#) guide. See also [Use AI Keyword to Enter Prompts](#) for more information on the Select AI actions.

Supported Classes

The library includes classes to manage providers, profiles, conversations, vector indexes, and synthetic data. Both synchronous and asynchronous versions are available.

- **Provider Classes:** Define AI provider: `OpenAIProvider`, `AzureProvider`, `OCI GenAIProvider`, `AWSProvider`, `GoogleProvider`, `AnthropicProvider`, `CohereProvider`, `HuggingFaceProvider`.
- **Profile:** defines the generative AI profile to process prompts (provider, credentials, metadata, options) and supports synthetic data generation.
- **ProfileAttributes:** profile configuration details such as provider, credential name, max tokens, temperature, object list, or vector index.
- **ConversationAttributes:** manage conversational context across prompts.
- **VectorIndex and VectorIndexAttributes:** create and manage vector indexes for RAG.
- **SyntheticDataAttributes:** create synthetic datasets for testing and development.

Asynchronous equivalents exist for `Profile`, `Conversation`, and `VectorIndex` classes.

For complete API reference, see [Select AI for Python](#) guide.

API and Attribute Update Enhancements

The following enhancements are available:

- All proxy objects now support `fetch()` to retrieve existing objects.
- All proxy objects provide `set_attribute()` and `set_attributes()` for consistent updates

Privilege and HTTP Access

Privilege management is separate from HTTP access configuration.

Privilege APIs:

- `select_ai.grant_privileges`
- `select_ai.revoke_privileges`

Privileges are updated to include:

- `DBMS_CLOUD`
- `DBMS_CLOUD_AI`
- `DBMS_CLOUD_PIPELINE`

HTTP access APIs:

- `select_ai.grant_http_access`
- `select_ai.revoke_http_access`

Other Enhancements

- Python 3.14 support
- New HTML documentation site on GitHub using the Python docs theme: [GitHub Select AI for Python Documentation](#)

Examples of Using Select AI

Explore integrating Oracle's Select AI with various supported AI providers to generate, run, and explain SQL from natural language prompts or chat with the LLM.

Example: Select AI Actions

The following example illustrates actions such as `runsql` (the default), `showsql`, `narrate`, `chat`, `explainsql`, `feedback`, and `summarize` that you can perform with `SELECT AI`. These examples use the `sh` schema with AI provider and profile attributes specified in the `DBMS_CLOUD_AI.CREATE_PROFILE` function. Use Select AI actions after setting your AI profile by using the `DBMS_CLOUD_AI.SET_PROFILE` procedure in the current session.

To generate a summary of your text, use `SELECT AI SUMMARIZE <TEXT>`.

```
SQL> select ai how many customers exist;
```

```
CUSTOMER_COUNT
-----
          55500
```

```
SQL> select ai showsql how many customers exist;
```

```
RESPONSE
-----
SELECT COUNT(*) AS total_customers
FROM SH.CUSTOMERS
```

```
SQL> select ai narrate how many customers exist;
```

```
RESPONSE
-----
There are a total of 55,500 customers in the database.
```

```
SQL> select ai chat how many customers exist;
```

```
RESPONSE
-----
--
It is impossible to determine the exact number of customers that exist as it
con
stantly changes due to various factors such as population growth, new
businesses
, and customer turnover. Additionally, the term "customer" can refer to
```

individuals, businesses, or organizations, making it difficult to provide a specific number.

```
SQL> select ai explainsql how many customers in San Francisco are married;
```

RESPONSE

```
-----
--
--
SELECT COUNT(*) AS customer_count
FROM SH.CUSTOMERS AS c
WHERE c.CUST_STATE_PROVINCE = 'San Francisco' AND c.CUST_MARITAL_STATUS =
'Married';
```

Explanation:

- We use the 'SH' table alias for the 'CUSTOMERS' table for better readability.
- The query uses the 'COUNT(*)' function to count the number of rows that match the given conditions.
- The 'WHERE' clause is used to filter the results:
 - 'c.CUST_STATE_PROVINCE = 'San Francisco'' filters customers who have 'San Francisco' as their state or province.
 - 'c.CUST_MARITAL_STATUS = 'Married'' filters customers who have 'Married' as their marital status.

The result of this query will give you the count of customers in San Francisco who are married, using the column alias 'customer_count' for the result.

Remember to adjust the table and column names based on your actual schema if they differ from the example.

Feel free to ask if you have more questions related to SQL or database in general.

```
-- Feedback on SQL Text
-- Negative feedback example:
SQL > select ai feedback for query "select ai showsql how many watch
histories in total", please use sum instead of count;
-- Positive feedback example:
SQL > select ai feedback for query "select ai showsql how many watch
histories in total", the sql query generated is correct;

-- Feedback on SQL ID
-- Negative feedback example:
SQL > select ai feedback please use sum instead of count for sql_id
1v1z68ra6r9zf;
-- Positive feedback example:
SQL > select ai feedback sql query result is correct for sql_id
1v1z68ra6r9zf;

-- If not specified, use default LASTAI SQL
-- To use default LASTAI sql, make sure that set server output off;
-- Negative feedback example:
SQL > select ai feedback please use ascending sorting for ranking;
```

```
-- Positive feedback example:
SQL > select ai feedback the result is correct;
```

```
SQL> SELECT AI SUMMARIZE
```

Like countless other people around the globe, I stream music, and like more than six hundred million of them I mainly use Spotify. Streaming currently accounts for about eighty per cent of the American recording industry's revenue, and in recent years Spotify's health is often consulted as a measure for the health of the music business over all. Last spring, the International Federation of the Phonographic Industry reported global revenues of \$28.6 billion, making for the ninth straight year of growth. All of this was unimaginable in the two-thousands, when the major record labels appeared poorly equipped to deal with piracy and the so-called death of physical media. On the consumer side, the story looks even rosier. Adjusted for inflation, a

... (skipped 1000 rows in the middle)

Pelly writes of some artists, in search of viral fame, who surreptitiously use social media to effectively beta test melodies and motifs, basically putting together songs via crowdsourcing. Artists have always fretted about the pressure to conform, but the data-driven, music-as-content era feels different. "You are a Spotify employee at that point," Daniel Lopatin, who makes abstract electronic music as Oneohtrix Point Never, told Pelly. "If your art practice is so ingrained in the brutal reality that Spotify has outlined for all of us, then what is the music that you're not making? What does the music you're not making sound like?" Listeners might wonder something similar. What does the music we're not hearing sound like?;

RESPONSE

The music streaming industry, led by Spotify, has revolutionized the way people consume music, with streaming accounting for 80% of the American recording industry's revenue. However, this shift has also complicated the lives of artists, who struggle to survive in a hyper-abundant present where music is often valued for its convenience rather than its artistic merit. Spotify's algorithms prioritize popularity and profitability over artistic diversity, leading to a homogenization of music and a devaluation of the labor that goes into creating it. Meanwhile, the company's executives reap enormous profits, with CEO Daniel Ek's net worth rivaling that of the wealthiest musicians. As music critic Liz Pelly argues, the streaming economy raises important questions about autonomy, creativity, and the value of art in a world where everything is readily available and easily accessible.

Example: Select AI with AWS

This example shows how you can use AWS to generate, run, and explain SQL from natural language prompts or chat using the models available with AWS.

The following example shows how to use AWS as the AI provider with Amazon Bedrock and its foundation models. The example shows creating AWS credentials, provide network access, creating an AI profile, and using Select AI actions to generate SQL queries from natural language prompts and chat using the AWS foundation models.

To use AWS, obtain access key, secret keys, and model ID. See [#unique_573](#). Use the model ID as the model attribute in the DBMS_CLOUD_AI.CREATE_PROFILE procedure. You must specify the model attribute explicitly, as no default model is provided.

```
--Grant EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--
-- Create Credential for AI provider
--
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'AWS_CRED',
        username       => '<your_AWS_access_key>',
        password       => '<your_AWS_secret_key>'
    );
END;
/
```

PL/SQL procedure successfully completed.

```
--
-- Grant Network ACL for AWS
--
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'bedrock-runtime.us-east-1.amazonaws.com',
        ace  => xs$ace_type(privilege_list => xs$name_list('http'),
            principal_name => 'ADB_USER',
            principal_type => xs_acl.ptype_db)
    );
END;
/
```

PL/SQL procedure successfully completed.

```
--
-- Create AI profile
--
BEGIN
    DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name => 'AWS',
        attributes   => {"provider": "aws",
            "credential_name": "AWS_CRED",
            "object_list": [{"owner": "SH", "name":
"customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}]},
        "model" : "anthropic.claude-v2",
        "conversation" : "true"
```



```

        }');
END;
/

```

PL/SQL procedure successfully completed.

```

--
-- Enable AI profile in current session
--
EXEC DBMS_CLOUD_AI.SET_PROFILE('AWS');

```

PL/SQL procedure successfully completed.

```

--
-- Use AI
--

```

```

SELECT AI how many customers exist;
"RESPONSE"
"COUNT(*)"
55500

```

```

SELECT AI how many customers in San Francisco are married;
"RESPONSE"
"COUNT(*)"
46

```

```

SELECT AI showsql how many customers in San Francisco are married;
"RESPONSE"
"SELECT COUNT(*) AS "Number of Married Customers in San Francisco"
FROM "SH"."CUSTOMERS" C
WHERE UPPER(C."CUST_CITY") = UPPER('San Francisco')
AND UPPER(C."CUST_MARITAL_STATUS") = UPPER('Married')"

```

```

SELECT AI explainsql how many customers in San Francisco are married;

```

```

"RESPONSE""SELECT
COUNT(*) AS "Number of Married Customers in San Francisco"
FROM "SH"."CUSTOMERS" C
WHERE C."CUST_CITY" = 'San Francisco'
AND C."CUST_MARITAL_STATUS" = 'Married'

```

Explanation:

- Used table alias C for CUSTOMERS table
- Used easy to read column names like CUST_CITY, CUST_MARITAL_STATUS
- Enclosed table name, schema name and column names in double quotes
- Compared string values in WHERE clause without UPPER() since the values are not in double quotes
- Counted number of rows satisfying the condition and aliased the count as "Number of Married Customers in San Francisco"

```

SELECT AI narrate what are the top 3 customers in San Francisco;
"RESPONSE"

```

The top 3 customers in San Francisco ordered by credit limit in descending order are:

1. Bert Katz
2. Madallyn Ladd
3. Henrietta Snodgrass

```
SELECT AI chat what is Autonomous AI Database;  
"RESPONSE"
```

"An Autonomous AI Database is a cloud database service provided by Oracle Corporation. Some key features of Oracle Autonomous AI Database include:

- Fully automated and self-driving - The database automatically upgrades, patches, tunes, and backs itself up without any human intervention required.
- Self-securing - The database uses machine learning to detect threats and automatically apply security updates.
- Self-repairing - The database monitors itself and automatically recovers from failures and errors without downtime.
- Self-scaling - The database automatically scales compute and storage resources up and down as needed to meet workload demands.
- Serverless - The database is accessed as a cloud service without having to manually provision any servers or infrastructure.
- High performance - The database uses Oracle's advanced automation and machine learning to continuously tune itself for high performance.
- Multiple workload support - Supports transaction processing, analytics, graph processing, etc in a single converged database.
- Fully managed - Oracle handles all the management and administration of the database. Users just load and access their data.
- Compatible - Supports common SQL and Oracle PL/SQL for easy migration from on-prem Oracle databases.

So in summary, an Oracle Autonomous AI Database is a fully automated, self-driving, self-securing, and self-repairing database provided as a simple cloud service. The automation provides high performance, elasticity, and availability with minimal human labor required."

```
--  
--Clear the profile  
--  
BEGIN  
    DBMS_CLOUD_AI.CLEAR_PROFILE;  
END;  
/  
PL/SQL procedure successfully completed.  
  
--  
--Drop the profile  
--
```

```
EXEC DBMS_CLOUD_AI.DROP_PROFILE('AWS');
```

PL/SQL procedure successfully completed.

Example: Enable Conversations in Select AI

These examples illustrates enabling conversations in Select AI.

Note

A user with administrator privileges (ADMIN) must grant EXECUTE and enable network access control list (ACL).

Session-Based Conversations

Create your AI profile. Set the conversation attribute to true in the profile, this action includes content from prior interactions or prompts, potentially including schema metadata, and set your profile. Once the profile is enabled, you can begin having conversations with your data. Use natural language to ask questions and follow up as needed.

```
--Grants EXECUTE privilege to ADB_USER
--
SQL> grant execute on DBMS_CLOUD_AI to ADB_USER;

-- Grant Network ACL for OpenAI endpoint
--
SQL> BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'api.openai.com',
        ace => xs$ace_type(privilege_list => xs$name_list('http'),
            principal_name => 'ADB_USER',
            principal_type => xs_acl.ptype_db)
    );
END;
/
```

PL/SQL procedure successfully completed.

```
--
-- Create Credential for AI provider
--
EXEC
DBMS_CLOUD.CREATE_CREDENTIAL(
CREDENTIAL_NAME => 'OPENAI_CRED',
username        => 'OPENAI',
password        => '<your_api_token>');
```

PL/SQL procedure successfully completed.

```
--
-- Create AI profile
--
BEGIN
```

```

DBMS_CLOUD_AI.CREATE_PROFILE(
  profile_name => 'OPENAI',
  attributes   => '{"provider":
"openai",
  "credential_name": "OPENAI_CRED",
  "object_list": [{"owner": "SH", "name": "customers"},
                  {"owner": "SH", "name": "countries"},
                  {"owner": "SH", "name": "supplementary_demographics"},
                  {"owner": "SH", "name": "profits"},
                  {"owner": "SH", "name": "promotions"},
                  {"owner": "SH", "name": "products"}],
  "conversation": "true"
}');

```

```

END;
/

```

PL/SQL procedure successfully completed.

```

--
-- Enable AI profile in current session
--
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE('OPENAI');

```

PL/SQL procedure successfully completed.

```

--
-- Get Profile in current session
--
SQL> SELECT DBMS_CLOUD_AI.get_profile() from dual;

```

```

DBMS_CLOUD_AI.GET_PROFILE()

```

```

-----
--
"OPENAI"

```

```

--
-- Use AI
--
what are the total number of customers;

```

```

CUSTOMER_COUNT

```

```

-----
55500

```

break out count of customers by country;

```

RESPONSE

```

```

-----
COUNTRY_NAME      CUSTOMER_COUNT
Italy              7780
Brazil             832
Japan              624
United Kingdom    7557
Germany            8173
United States of America 18520

```

France	3833
Canada	2010
Spain	2039
China	712
Singapore	597
New Zealand	244
Poland	708
Australia	831
Argentina	403
Denmark	383
South Africa	88
Saudi Arabia	75
Turkey	91

what age group is most common;

RESPONSE

```
-----
--
AGE_GROUP      CUSTOMER_COUNT
65+            28226
```

select ai keep the top 5 customers and their country by their purchases and include a rank in the result;

RESPONSE

```
-----
--
RANK      CUSTOMER_NAME      COUNTRY      PURCHASES
1         Abigail Ruddy      Japan         276
2         Abigail Ruddy      Italy         168
3         Abigail Ruddy      Japan         74
3         Abner Robbinette    Germany       74
5         Abner Everett       France        68
```

```
SQL> EXEC DBMS_CLOUD_AI.DROP_PROFILE('OPENAI');
```

PL/SQL procedure successfully completed.

Customizable Conversations

The following examples demonstrate using the conversation management API supporting customizable conversations. To use Select AI for multiple conversations:

1. Create a conversation
2. Set the conversation in the current user session
3. Use Select AI <action> <prompt>
 - Use `DBMS_CLOUD_AI.CREATE_CONVERSATION` function and then set the conversation using `DBMS_CLOUD_AI.SET_CONVERSATION_ID`.
 - Call the `DBMS_CLOUD_AI.CREATE_CONVERSATION` procedure directly to create and set the conversation in one step.

Example: Create and Set Customizable Conversations

The following example demonstrates how to create a conversation using DBMS_CLOUD_AI.CREATE_CONVERSATION function and set it using the DBMS_CLOUD_AI.SET_CONVERSATION_ID procedure.

```
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION; -- in 19c, run SELECT
DBMS_CLOUD_AI.create_conversation FROM dual;
```

```
CREATE_CONVERSATION
-----
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92
```

```
EXEC DBMS_CLOUD_AI.SET_CONVERSATION_ID('30C9DB6E-EA4D-AFBA-
E063-9C6D46644B92');
```

PL/SQL procedure successfully completed

The following example demonstrates running the DBMS_CLOUD_AI.CREATE_CONVERSATION procedure to create and set the conversation_id directly.

```
EXEC DBMS_CLOUD_AI.create_conversation;
```

PL/SQL procedure successfully completed.

You can also customize the conversation attributes such as title, description, retention_days, and conversation_length attributes.

```
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION(
      attributes => '{"title":"My first conversation",
                    "description":"this is my first conversation",
                    "retention_days":5,
                    "conversation_length":5}');

```

```
CREATE_CONVERSATION
-----
38F8B874-7687-2A3F-E063-9C6D4664EC3A
```

You can view if a certain conversation exists by querying DBA/USER_CLOUD_AI_CONVERSATIONS view.

```
-- Verify the setup
SELECT conversation_id, conversation_title, description, retention_days,
       conversation_length FROM DBA_CLOUD_AI_CONVERSATIONS WHERE
       conversation_id = '38F8B874-7687-2A3F-E063-9C6D4664EC3A';
```

```
CONVERSATION_ID
CONVERSATION_TITLE
DESCRIPTION                                RETENTION_DAYS
CONVERSATION_LENGTH
-----
-----
-----
```

```

-----
38F8B874-7687-2A3F-E063-9C6D4664EC3A      My first
conversation                                this is my first conversation
+000005 00:00:00.000000                      5

```

You can also verify if a conversation is set by calling the `DBMS_CLOUD_AI.GET_CONVERSATION_ID` function.

```
SELECT DBMS_CLOUD_AI.GET_CONVERSATION_ID;
```

```

-----
--
30C9DB6E-EA4F-AFBA-E063-9C6D46644B92

```

Example: Use Customizable Conversations with Select AI

After you create and set the conversation and enable your AI profile, you can start interacting with your data. Use natural language to ask questions and follow up as needed.

Use `SELECT AI <ACTION> <PROMPT>`.

```
SELECT AI CHAT What is the difference in weather between Seattle and San
Francisco?;
```

RESPONSE

```

-----
--
Seattle and San Francisco are both located on the West Coast of the United
State
s, but they have distinct weather patterns due to their unique geography and
cli
mate conditions. Here are the main differences:

1. **Rainfall**: Seattle is known for its rainy reputation, with an average
annu
al rainfall of around 37 inches (94 cm). San Francisco, on the other hand,
recei
ves significantly less rainfall, with an average of around 20 inches (51 cm)
per
year.
2. **Cloud Cover**: Seattle is often cloudy, with an average of 226 cloudy
days
per year. San Francisco is also cloudy, but to a lesser extent, with an
average
of 165 cloudy days per year.

.....

```

```
SELECT AI CHAT Explain the difference again in one paragraph only.;
```

RESPONSE

```

-----
--
Seattle and San Francisco have different weather patterns despite both

```

experiencing a mild oceanic climate. San Francisco tends to be slightly warmer, with average temperatures ranging from 45°F to 67°F, and receives less rainfall, around 20 inches per year, mostly during winter. In contrast, Seattle is cooler, with temperatures ranging from 38°F to 64°F, and rainier, with around 37 inches of rainfall per year, distributed throughout the year. San Francisco is also known for its fog, particularly during summer, and receives more sunshine, around 160 sunny days per year, although it's often filtered through the fog. Overall, San Francisco's weather is warmer and sunnier, with more pronounced seasonal variations, while Seattle's is cooler and rainier, with more consistent temperatures throughout the year.

Example: Use GENERATE Function to Compare Two Conversations

The following example shows how two conversations are used interchangeably to ask questions and verify accurate responses. Each conversation begins with a different question focused on comparison. Later, when you ask the same follow-up question in both conversations, each returns a different answer based on its prior context.

```
-- First conversation
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION;

CREATE_CONVERSATION
-----
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92

-- Second conversation
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION;

CREATE_CONVERSATION
-----
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92

-- Call generate using the first conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
    prompt      => 'What is the difference in weather between Seattle
and San Francisco?',
    profile_name => 'GENAI',
    action      => 'CHAT',
    params      => '{"conversation_id":"30C9DB6E-EA4D-AFBA-
E063-9C6D46644B92"}') AS RESPONSE;
```


RESPONSE

```
-----
--
Seattle and San Francisco, both located in the Pacific Northwest and Northern
California respectively, experience a mild oceanic climate. However, there are
some notable differences in their weather patterns:

1. **Temperature**: San Francisco tends to be slightly warmer than Seattle,
especially during the summer months. San Francisco's average temperature ranges
from 45°F (7°C) in winter to 67°F (19°C) in summer, while Seattle's average
temperature ranges from 38°F (3°C) in winter to 64°F (18°C) in summer.

2. **Rainfall**: Seattle is known for its rainy reputation, with an average
annual rainfall of around 37 inches (94 cm). San Francisco receives less
rainfall, with an average of around 20 inches (51 cm) per year. However, San Francisco's
rainfall is more concentrated during the winter months, while Seattle's
rainfall is more evenly distributed throughout the year.

.....

-- Call generate using the second conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
    prompt      => 'How does the cost of living compare between New
York and Los Angeles?',
    profile_name => 'GENAI',
    action       => 'CHAT',
    params       => '{"conversation_id":"30C9DB6E-EA4E-AFBA-
E063-9C6D46644B92"}') AS RESPONSE;
```

RESPONSE

```
-----
--
The cost of living in New York and Los Angeles is relatively high compared to
other cities in the United States. However, there are some differences in the
cost of living between the two cities. Here's a comparison of the cost of living
in New York and Los Angeles:

1. Housing: The cost of housing is significantly higher in New York than in
Los Angeles. The median home price in New York is around $999,000, while in Los
Angeles it's around $849,000. Rent is also higher in New York, with the average
```

rent
for a one-bedroom apartment being around \$3,000 per month, compared to
around \$
2,400 per month in Los Angeles.

2. Food: The cost of food is relatively similar in both cities, with some
variat
ion in the cost of certain types of cuisine. However, eating out in New York
can
be more expensive, with the average cost of a meal at a mid-range restaurant
be
ing around \$15-20 per person, compared to around \$12-18 per person in Los
Angele
s.

.....

```
-- Call generate using the first conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
  prompt      => 'Explain the difference again in one paragraph
only.',
  profile_name => 'GENAI',
  action       => 'CHAT',
  params       => '{"conversation_id":"30C9DB6E-EA4D-AFBA-
E063-9C6D46644B92"}') AS RESPONSE;
```

RESPONSE

```
-----
--
Seattle and San Francisco have different weather patterns despite both
experienc
ing a mild oceanic climate. San Francisco tends to be slightly warmer, with
aver
age temperatures ranging from 45°F to 67°F, and receives less rainfall,
around 2
0 inches per year, mostly during winter. In contrast, Seattle is cooler, with
te
mperatures ranging from 38°F to 64°F, and rainier, with around 37 inches of
rain
fall per year, distributed throughout the year. San Francisco is also known
for
its fog, particularly during summer, and receives more sunshine, around 160
sunn
y days per year, although it's often filtered through the fog. Overall, San
Fran
cisco's weather is warmer and sunnier, with more pronounced seasonal
variations,
while Seattle's is cooler and rainier, with more consistent temperatures
throug
hout the year.
```

```
-- Call generate using the second conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
```

```

        prompt      => 'Explain the difference again in one paragraph
only.',
        profile_name => 'GENAI',
        action       => 'CHAT',
        params       => '{"conversation_id":"30C9DB6E-EA4E-AFBA-
E063-9C6D46644B92"}') AS RESPONSE;

```

RESPONSE

```

-----
--
The cost of living in New York is approximately 20-30% higher than in Los
Angeles, mainly due to the higher cost of housing and transportation. New York has
a median home price of around $999,000 and average rent of $3,000 per month for
a one-bedroom apartment, compared to Los Angeles' median home price of $849,000
and average rent of $2,400 per month. While the cost of food and utilities is
relatively similar in both cities, the cost of transportation is higher in Los
Angeles due to its car-centric culture, but the cost of public transportation is
higher in New York. Overall, the total monthly expenses for a single person in New
York can range from $4,600, compared to around $4,050 in Los Angeles, making
New York the more expensive city to live in.

```

You may call the `DBMS_CLOUD_AI.GENERATE` function without specifying a conversation; however, in such cases, a meaningful response should not be expected.

```

-- Ask SELECT AI using the second conversation.
SELECT DBMS_CLOUD_AI.GENERATE(
        prompt      => 'Explain the difference again in one paragraph
only.',
        profile_name => 'GENAI',
        action       => 'CHAT') AS RESPONSE;

```

RESPONSE

```

-----
--
There is no previous explanation to draw from, as this is the beginning of
our conversation. If you would like to ask a question or provide a topic, I would
be happy to explain the differences related to it in one paragraph.

```

Example: Verify Conversations through DBMS_CLOUD_AI Views

You can query the `DBMS_CLOUD_AI` conversation views to review conversation and prompt details. See [#unique_574](#) for more details.

Note

The Views with the DBA_ prefix are available only to users with administrator privileges (ADMIN).

```
SELECT conversation_id, conversation_title, description FROM
dba_cloud_ai_conversations;
```

```
CONVERSATION_ID
```

```
CONVERSATION_TITLE
```

```
DESCRIPTION
```

```
--
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92
Seattle vs San Francisco Weather
The conversation discusses the comparison of weather patterns between Seattle
an
d San Francisco, focusing on the differences in temperature, rainfall, fog,
suns
hine, and seasonal variation between the two cities.
```

```
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92
NY vs LA Cost Comparison
The conversation discusses and compares the cost of living in New York and
Los A
ngeles, covering housing, food, transportation, utilities, and taxes to
provide
an overall view of the expenses in both cities.
```

```
SELECT conversation_id, count(*) FROM dba_cloud_ai_conversation_prompts
GROUP BY conversation_id;
```

```
CONVERSATION_ID          COUNT(*)
-----
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92      2
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92      2
```

Example: Update Conversation Details

You can update the title, description, and retention_days of a conversation using the DBMS_CLOUD_AI.UPDATE_CONVERSATION procedure. You can verify the update by querying the DBMS_CLOUD_AI conversation view.

```
-- Update the second conversation's title, description and retention_days
SQL> EXEC DBMS_CLOUD_AI.update_conversation(conversation_id => '30C9DB6E-EA4E-
AFBA-E063-9C6D46644B92',
attributes =>
'{"retention_days":20,
"description":"This a
description",
```

```
"title":"a title",
```

```
"conversation_length":20}');
```

PL/SQL procedure successfully completed.

```
-- Verify the information for the second conversation
SQL> SELECT conversation_title, description, retention_days
FROM dba_cloud_ai_conversations
WHERE conversation_id = '30C9DB6E-EA4E-AFBA-E063-9C6D46644B92';
```

CONVERSATION_TITLE	DESCRIPTION
RETENTION_DAYS	LENGTH
-----	-----
a title	This a description
20	20

Example: Delete a Prompt

You can delete an individual prompt from your conversations and verify the modification by querying the DBMS_CLOUD_AI conversation view.

```
-- Find the latest prompt for first conversation
SELECT conversation_prompt_id FROM dba_cloud_ai_conversation_prompts
WHERE conversation_id = '30C9DB6E-EA4D-AFBA-E063-9C6D46644B92'
ORDER BY created DESC
FETCH FIRST ROW ONLY;
```

CONVERSATION_PROMPT_ID

30C9DB6E-EA61-AFBA-E063-9C6D46644B92

```
-- Delete the prompt
EXEC DBMS_CLOUD_AI.DELETE_CONVERSATION_PROMPT('30C9DB6E-EA61-AFBA-
E063-9C6D46644B92');
```

PL/SQL procedure successfully completed.

```
-- Verify if the prompt is deleted
SELECT conversation_prompt_id FROM dba_cloud_ai_conversation_prompts
WHERE conversation_id = '30C9DB6E-EA4D-AFBA-E063-9C6D46644B92';
```

CONVERSATION_PROMPT_ID

30C9DB6E-EA5A-AFBA-E063-9C6D46644B92

Example: Drop a Conversation

You can delete the entire conversation, which also removes all prompts associated with it.

```
-- Delete the first conversation
EXEC DBMS_CLOUD_AI.DROP_CONVERSATION('30C9DB6E-EA4D-AFBA-E063-9C6D46644B92');
```

PL/SQL procedure successfully completed.

```
-- Verify if the conversation and its prompts are removed
SELECT conversation_id FROM dba_cloud_ai_conversations;
```

```
-- We only have the second conversation now
CONVERSATION_ID
-----
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92
```

```
SELECT conversation_id, count(*) FROM dba_cloud_ai_conversation_prompts GROUP
BY conversation_id;
```

```
-- We only have prompts in the second conversation
CONVERSATION_ID          COUNT(*)
-----
30C9DB6E-EA4E-AFBA-E063-9C6D46644B92          2
```

Example: Select AI Feedback

These examples demonstrate how you can use the `DBMS_CLOUD_AI.FEEDBACK` procedure and the different scenarios for providing feedback to improve subsequent SQL query generation.

Note

You can provide feedback for Select AI SQL prompts even if the prompt has not been run previously. Select AI does not require the SQL prompt to be used in a query before submitting feedback; you may supply feedback for any valid prompt at any time.

Example: Provide Negative Feedback

The following example demonstrates providing corrections to the generated SQL as feedback (negative feedback) using `feedback_type` as *negative* and providing your SQL query.

You add your feedback to the AI profile named `OCI_FEEDBACK1` by calling the `DBMS_CLOUD_AI.FEEDBACK` procedure with the `sql_text` parameter containing the prompt. See [#unique_575](#) to learn about the attributes. Then, you retrieve the content and attributes columns from the `<profile_name>_FEEDBACK_VECINDEX$VECTAB` table, which is linked to that specific SQL query. Select AI automatically creates this vector table when you first use the feedback feature. See [#unique_576](#) for more information.

```
SQL> select ai showsql how many movies;
```

RESPONSE

```
-----
SELECT COUNT(m."MOVIE_ID") AS "Number of Movies" FROM "ADB_USER"."MOVIES" m
```

```
SQL> exec DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1', sql_text=>
'select ai showsql how many movies', feedback_type=> 'negative',
response=>'SELECT SUM(1) FROM "ADB_USER"."MOVIES"');
```

PL/SQL procedure successfully completed.

```
SQL> select CONTENT, ATTRIBUTES from OCI_FEEDBACK1_FEEDBACK_VECINDEX$VECTAB
where JSON_VALUE(attributes, '$.sql_text') = 'select ai showsql how many
movies';
```

```
CONTENT
```

```
-----
```

```
how many movies
```

```
ATTRIBUTES
```

```
-----
```

```
{"response":"SELECT SUM(1) FROM
\"ADB_USER\".\"MOVIES\"\",\"feedback_type\":\"negative\",\"sql_id\":null,\"sql_text\":\"
select ai showsql how many movies\",\"feedback_content\":null}
```

Example: Provide Positive Feedback

The following example demonstrates providing your approval that you agree and confirm the generated SQL (positive feedback) using `feedback_type` as *positive*.

In this example, the query retrieves the `sql_id` from the `v$mapped_sql` view for the given prompt. See `V_MAPPED_SQL` for more information.

You add your feedback to the AI profile named `OCI_FEEDBACK1` by calling the `DBMS_CLOUD_AI.FEEDBACK` procedure with the `sql_id` parameter. Then, you retrieve the content and attributes columns from the `<profile_name>_FEEDBACK_VECINDEX$VECTAB` table, which is linked to that specific SQL query. Select AI automatically creates this vector table when you first use the feedback feature. See [#unique_576](#) for more information.

```
SQL> select ai showsql how many distinct movie genres?;
```

```
RESPONSE
```

```
-----
```

```
SELECT COUNT(DISTINCT g."GENRE_NAME") AS "Number of Movie Genres" FROM
"ADB_USER"."GENRES" g
```

```
SQL> SELECT sql_id FROM v$mapped_sql WHERE sql_text = 'select ai showsql how
many distinct movie genres?';
```

```
SQL_ID
```

```
-----
```

```
852w8u83gk1
```

```
SQL> exec DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1', sql_id=>
'852w8u83gk1', feedback_type=>'positive', operation=>'add');
```

PL/SQL procedure successfully completed.

```
SQL> SELECT content, attributes FROM OCI_FEEDBACK1_FEEDBACK_VECINDEX$VECTAB
WHERE JSON_VALUE(attributes, '$.sql_id') = '852w8u83gk1';
```

```
CONTENT
```

```
-----
```

```
how many distinct movie genres?
```

ATTRIBUTES

```

-----
{"response": "SELECT COUNT(DISTINCT g.\"GENRE_NAME\") AS \"Number of Movie Genres\" FROM \"ADB_USER\".\"GENRES\" g", "feedback_type": "positive", "sql_id": "852w8u83gk1", "sql_text": "select ai showsql how many distinct movie genres?", "feedback_content": null}

```

Example: Provide Feedback Without Prior Usage

You may provide feedback for SQL prompts even if the prompt has not been used previously. For example:

```

BEGIN
  DBMS_CLOUD_AI.FEEDBACK(
    profile_name=>'AI_PROFILE',
    sql_text=>'select ai runsql how many products named PAD', -- Prior usage
    not required
    feedback_type=>'negative',
    response=>'SELECT COUNT(*) AS "Num" FROM "PRODUCTS"."CATG" o WHERE UPPER(o."NAME") LIKE '%PAD%',
    feedback_content=>'Use LIKE instead of ='
  );
END;
/

```

In this case, feedback for the prompt `select ai runsql how many products named PAD` is submitted without needing to use the prompt beforehand.

Example: Add or Delete Your Feedback for the Generated SQL

The following example demonstrates adding or deleting your feedback for the generated SQL by specifying the `DBMS_CLOUD_AI.FEEDBACK` procedure parameters. This example demonstrates using `sql_id` and `sql_text` along with other parameters.

Note

Select AI allows only a single feedback entry for each `sql_id`. If you provide additional feedback for the same `sql_id`, Select AI replaces the previous entry with the new one.

See [#unique_575](#) for more details on the parameters.

```

EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',
                             sql_id=>'852w8u83gk1',
                             feedback_type=>'positive',
                             operation=>'add');
EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',
                             sql_text=>'select ai showsql how many
movies',
                             feedback_type=>'negative',
                             response=>'SELECT SUM(1) FROM
"ADB_USER"."MOVIES"',
                             feedback_content=>'Use SUM instead of
COUNT');
EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',

```



```
sql_id=> '852w8u83gk1',
operation=>'delete');
```

Example: Use Feedback Action with the Last AI SQL to Provide Negative Feedback

This example demonstrates using feedback action to improve the generated SQL by suggesting the modifications using natural language.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name=>'OCI_FEEDBACK1',
    attributes=> '{"provider":
"oci",
  "credential_name": "GENAI_CRED",
  "oci_compartment_id": "ocid1.compartment.oc1..aaaa...",
  "object_list": [{ "owner": "ADB_USER", "name": "users"},
    { "owner": "ADB_USER", "name": "movies"},
    { "owner": "ADB_USER", "name": "genres"},
    { "owner": "ADB_USER", "name": "watch_history"},
    { "owner": "ADB_USER", "name": "movie_genres"},
    { "owner": "ADB_USER", "name": "employees1"},
    { "owner": "ADB_USER", "name": "employees2"}
  ]
  }');
END;
/

EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_FEEDBACK1');
```

PL/SQL procedure successfully completed.

```
select ai showsql rank movie duration;
```

RESPONSE

```
-----
-
SELECT "DURATION" AS "Movie Duration" FROM "ADB_USER"."MOVIES" ORDER BY
"DURATION"
```

```
select ai feedback use ascending sorting;
```

RESPONSE

```
-----
-----
-----
Based on your feedback, the SQL query for prompt "rank movie duration" is
successfully refined. The refined SQL query as following:
SELECT m."DURATION" AS "Movie Duration" FROM "ADB_USER"."MOVIES" m ORDER BY
m."DURATION" ASC
```

```
select ai showsql rank the movie duration;
```

RESPONSE

```
-----
-----
SELECT m."DURATION" AS "Movie Duration" FROM "ADB_USER"."MOVIES" m ORDER BY
m."DURATION" ASC
```

Example: Use Feedback Action with the Last AI SQL to Provide Positive Feedback

This example demonstrates using the feedback action to accept the generated SQL using natural language.

```
--Positive feedback
```

```
select ai showsql which movies are comedy?;
```

```
RESPONSE
```

```
-----
-----
SELECT DISTINCT m."TITLE" AS "Movie Title" FROM "ADB_USER"."MOVIES" m INNER
JOIN "ADB_USER"."MOVIE_GENRES" mg ON m."MOVIE_ID" = mg."MOVIE_ID" INNER JOIN
"ADB_USER"."GENRES" g ON mg."GENRE_ID" = g."GENRE_ID" WHERE g."GENRE_NAME" =
'comedy'
```

```
select ai feedback this is correct;
```

```
RESPONSE
```

```
-----
-----
Thank you for your positive feedback. The SQL query for prompt "which movies
are comedy?" is correctly implemented and delivering the expected results. It
will be referenced for future optimizations and improvements.
```

```
Select AI Feedback Action Referring SQL_ID
```

Example: Use Feedback Action with SQL_ID to Provide Feedback

This example demonstrates using SQL_ID with the feedback action to provide feedback for a particular generated SQL query. You can obtain the SQL_ID by querying the V\$MAPPED_SQL table.

```
-- Query mentioned with SQL_ID
```

```
select ai showsql how many movies are in each genre;
```

```
RESPONSE
```

```
-----
-----
SELECT g."GENRE_NAME" AS "Genre Name", COUNT(m."MOVIE_ID") AS "Number of
Movies" FROM "ADB_USER"."MOVIES" m INNER JOIN "ADB_USER"."MOVIE_GENRES" mg ON
m."MOVIE_ID" = mg."MOVIE_ID" INNER JOIN "ADB_USER"."GENRES" g ON
mg."GENRE_ID" = g."GENRE_ID" GROUP BY g."GENRE_NAME"
```

```
select sql_id from v$sql where sql_text = 'select ai showsql how
many movies are in each genre';
```

SQL_ID

8azkwc0hr87ga

select ai feedback for query with sql_id = '8azkwc0hr87ga', rank in descending sorting;

RESPONSE

Based on your feedback, the SQL query for prompt "how many movies are in each genre" is successfully refined. The refined SQL query as following:

```
SELECT g."GENRE_NAME" AS "Genre Name", COUNT(m."MOVIE_ID") AS "Number of
Movies"
FROM "ADB_USER"."MOVIES" m
INNER JOIN "ADB_USER"."MOVIE_GENRES" mg ON m."MOVIE_ID" = mg."MOVIE_ID"
INNER JOIN "ADB_USER"."GENRES" g ON mg."GENRE_ID" = g."GENRE_ID"
GROUP BY g."GENRE_NAME"
ORDER BY COUNT(m."MOVIE_ID") DESC
```

Example: Use Feedback Action with Query Text

This example shows the feedback action for a specific Select AI query by including the Select AI prompt in quotes followed by your feedback.

-Query mentioned with SQL_TEXT

select ai showsql how many watch history in total;

RESPONSE


```
SELECT COUNT(w."WATCH_ID") AS "Total Watch History" FROM
"ADB_USER"."WATCH_HISTORY" w
```

select ai feedback for query "select ai showsql how many watch history in total", name the column as total_watch;

RESPONSE

Based on your feedback, the SQL query for prompt "how many watch history in total" is successfully refined. The refined SQL query as following:

```
SELECT COUNT(w."WATCH_ID") AS "total_watch" FROM "ADB_USER"."WATCH_HISTORY" w
```

Example: Select AI for Property Graphs

This example shows how you can use the DBMS_CLOUD_AI.GENERATE procedure and a natural language prompt to generate PGQ graph queries to query graph data.

Example: Specify Multiple Graphs in your AI Profile

This example shows how to define multiple property graphs in your AI profile, including a sample query and its output.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'OPENAI',
    attributes  => '{"provider": "openai",
                  "model": "gpt-4o",
                  "credential_name": "OPENAI_CRED",
                  "object_list": [{"owner": "ADB_USER", "name": "LDBC_GRAPH"},
                                {"owner": "ADB_USER", "name": "G"}]
    ');
END;
/
```

```
SQL> EXEC DBMS_CLOUD_AI.SET_PROFILE(profile_name => 'OPENAI');
```

PL/SQL procedure successfully completed.

```
SQL> select ai who bought a dress;
```

```
CUSTOMER_N
-----
Kate
```

1 row selected.

Example: Sample Prompts for Property Graphs

These examples show how to create sample data and display the generated SQL using the `showsql` action for a given prompt.

Example: Create Property Graph Table

The following example creates sample tables and a property graph.

```
CREATE TABLE Person
(
  id          NUMBER PRIMARY KEY,
  firstName   VARCHAR2(20 CHAR),
  lastName    VARCHAR2(20 CHAR),
  age         NUMBER,
  jsonProp    VARCHAR2(40 CHAR)
);

CREATE TABLE Post
(
  id          NUMBER PRIMARY KEY,
  content     VARCHAR2(20 CHAR)
);

CREATE TABLE personLikesPost
(
  idPerson    NUMBER REFERENCES Person (id),
  idPost      NUMBER REFERENCES Post (id)
);
```

```

CREATE TABLE personKnowsPerson
(
    idPerson1 NUMBER REFERENCES Person (id),
    idPerson2 NUMBER REFERENCES Person (id)
);

CREATE PROPERTY GRAPH person_graph
  VERTEX TABLES (
    Person KEY (id) LABEL Person
      PROPERTIES (firstName, lastName, age, jsonProp),
    Post KEY (id) LABEL Post
      PROPERTIES(content)
  )
  EDGE TABLES (
    personLikesPost
      KEY(idPerson, idPost)
      SOURCE KEY (idPerson) REFERENCES Person (id)
      DESTINATION KEY (idPost) REFERENCES POST (id)
      LABEL Likes NO PROPERTIES,
    personKnowsPerson
      KEY(idPerson1, idPerson2)
      SOURCE KEY (idPerson1) REFERENCES Person (id)
      DESTINATION KEY (idPerson2) REFERENCES Person (id)
      LABEL Knows NO PROPERTIES
  );

insert into Person values (1, 'John', 'Doe', 23,
  '{"key1":"value1","key2":"value2"}');
insert into Person values (2, 'Scott', 'Tiger', 25,
  '{"key1":"value3","key2":"value4"}');
insert into Person values (3, 'Max', 'Power', 27,
  '{"key1":"value5","key2":"value6"}');
insert into Person values (4, 'Jane', 'Doe', 22,
  '{"key1":"value7","key2":"value8"}');
insert into Person (id, Firstname, age) values (5, 'Hans', 23);
insert into Person (id, Firstname, age) values (6, 'Franz', 24);

INSERT INTO Post VALUES (10, 'Lorem ipsum...');
INSERT INTO Post VALUES (11, 'Nulla facilisi...');
INSERT INTO Post VALUES (12, 'Vestibulum eget ...');
INSERT INTO Post VALUES (13, 'Sed fermentum...');
INSERT INTO Post VALUES (14, 'Fusce at ...');
INSERT INTO Post VALUES (15, 'Pellentesque sit ...');
INSERT INTO Post VALUES (16, 'Integer...');
INSERT INTO Post VALUES (17, 'Curabitur luctus ...');
INSERT INTO Post VALUES (18, 'Nam in ...');
INSERT INTO Post VALUES (19, 'Etiam ac ...');

insert into personKnowsPerson values (1, 2);
insert into personKnowsPerson values (2, 3);
insert into personKnowsPerson values (3, 4);
insert into personKnowsPerson values (4, 5);
insert into personKnowsPerson values (5, 6);
insert into personKnowsPerson values (6, 2);
insert into personKnowsPerson values (5, 3);

```

```

INSERT INTO personLikesPost VALUES (1, 10);
INSERT INTO personLikesPost VALUES (1, 11);
INSERT INTO personLikesPost VALUES (1, 12);
INSERT INTO personLikesPost VALUES (2, 10);
INSERT INTO personLikesPost VALUES (2, 13);
INSERT INTO personLikesPost VALUES (2, 14);
INSERT INTO personLikesPost VALUES (3, 11);
INSERT INTO personLikesPost VALUES (3, 15);
INSERT INTO personLikesPost VALUES (3, 16);
INSERT INTO personLikesPost VALUES (4, 12);
INSERT INTO personLikesPost VALUES (4, 17);
INSERT INTO personLikesPost VALUES (4, 18);
INSERT INTO personLikesPost VALUES (5, 13);
INSERT INTO personLikesPost VALUES (5, 14);
INSERT INTO personLikesPost VALUES (5, 19);
INSERT INTO personLikesPost VALUES (6, 15);
INSERT INTO personLikesPost VALUES (6, 16);
INSERT INTO personLikesPost VALUES (6, 17);
INSERT INTO personLikesPost VALUES (1, 18);
INSERT INTO personLikesPost VALUES (2, 19);

```

```
commit;
```

Example: Matching of Vertices Without Labels

Prompt: Find all the people IDs

```

SELECT person_id
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
    MATCH (v IS "PERSON")
    COLUMNS (VERTEX_ID(v) AS person_id))

```

Example: Matching of Edges Without Labels

Prompt: Find all the edge IDs

```

SELECT edge_id
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
    MATCH (v1) -[e]-> (v2)
    COLUMNS (EDGE_ID(e) AS edge_id))

```

Example: Matching of Edges With Labels

Prompt: Find all the knows relationship IDs

```

SELECT knows_id
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
    MATCH (p1 IS "PERSON") -[e IS "KNOWS"]-> (p2 IS "PERSON")
    COLUMNS (EDGE_ID(e) AS knows_id))

```

Example: Matching of Path Patterns Including One or Multiple Hops

Prompt: List all people who know someone who liked the post 'Sed fermentum...'

```

SELECT person_id, person_firstname, person_lastname
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
    MATCH (p1 IS "PERSON") -[e1 IS "KNOWS"]-> (p2 IS "PERSON") -[e2 IS

```

```
"LIKES"]-> (post IS "POST")
  WHERE UPPER(post."CONTENT") = UPPER('Sed fermentum...')
  COLUMNS (VERTEX_ID(p1) AS person_id, p1."FIRSTNAME" AS person_firstname,
p1."LASTNAME" AS person_lastname))
```

Example: Accessing Vertex and Edge Properties

Prompt: Find first name and last name of all people

```
SELECT first_name, last_name
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
  MATCH (p IS "PERSON")
  COLUMNS (p."FIRSTNAME" AS first_name, p."LASTNAME" AS last_name))
```

Example: Filtering data

Prompt: Find post contents liked by John Doe

```
SELECT post_content
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
  MATCH (p IS "PERSON") -[e IS "LIKES"]-> (post IS "POST")
  WHERE UPPER(p."FIRSTNAME") = UPPER('John') AND UPPER(p."LASTNAME") =
UPPER('Doe')
  COLUMNS (post."CONTENT" AS post_content))
```

Example: Functions and Expressions

Prompt: Show all people (full name) and display their key1 value from jsonProp property

```
SELECT person_fullname, json_key1
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
  MATCH (p IS "PERSON")
  COLUMNS (
    (p."FIRSTNAME" || ' ' || p."LASTNAME") AS person_fullname,
    JSON_QUERY(p."JSONPROP", '$.key1') AS json_key1
  )
)
```

Example: Sorting Data

Prompt: Find friends of Scott Tiger ordered by their last name

```
SELECT friend_firstname, friend_lastname
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
  MATCH (p IS "PERSON") -[e IS "KNOWS"]-> (f IS "PERSON")
  WHERE UPPER(p."FIRSTNAME") = UPPER('Scott') AND UPPER(p."LASTNAME") =
UPPER('Tiger')
  COLUMNS (f."FIRSTNAME" AS friend_firstname, f."LASTNAME" AS
friend_lastname)
)
ORDER BY friend_lastname
```

Example: Row Limiting

Prompt: Find all people ordered by first name. Skip one result and return 2 results only

```
SELECT person_firstname
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
    MATCH (p IS "PERSON")
    COLUMNS (p."FIRSTNAME" AS person_firstname))
ORDER BY person_firstname
OFFSET 1 ROWS FETCH NEXT 2 ROWS ONLY
```

Unsupported Queries

For the following queries certain LLMs generate valid NL2SQL, but the resulting SQL uses features that are not yet supported in Oracle AI Database 26ai.

Example: Queries Requiring not to Match a Certain Pattern

Prompt: Find people that do not know Scott.

EXISTS subquery is not supported.

```
SELECT person_id, first_name, last_name
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
    MATCH (p1 IS "PERSON")
    WHERE NOT EXISTS (
        SELECT 1
        FROM GRAPH_TABLE("ADB_USER"."PERSONGRAPH"
            MATCH (p2 IS "PERSON") -[e IS "PERSONKNOWSPERSON"]-> (p3 IS "PERSON"
        )
        WHERE p2."ID" = p1."ID" AND UPPER(p3."FIRSTNAME") = UPPER('Scott')
        COLUMNS (1 AS dummy_value))
    )
    COLUMNS (p1."ID" AS person_id, p1."FIRSTNAME" AS first_name,
p1."LASTNAME" A
S last_name))
```

Example: Queries Requiring to Optionally Match a Certain Pattern

Prompt: Show all people and how many posts they have liked (show people even if they have not liked a post).

OPTIONAL match is not supported.

```
SELECT person_id, person_firstname, person_lastname, liked_post_ids
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
    MATCH (p is "PERSON") OPTIONAL MATCH (p) -[l is "PERSONLIKESPOST"]-> (post is
"POST")
    COLUMNS(
        VERTEX_ID(p) as person_id,
        p."FIRSTNAME" as person_firstname,
        p."LASTNAME" as person_lastname,
        JSON_ARRAYAGG(VERTEX_ID(post)) as liked_post_ids
    ))
```

Example: Unbounded Recursive Path Patterns

Prompt: Find all people that Scott can reach.

Queries that use unbounded quantifiers are not supported.

```
SELECT person_id, person_firstname, person_lastname
FROM GRAPH_TABLE("ADB_USER"."PERSONGRAPH"
  MATCH (src IS "PERSON") -[e IS "PERSONKNOWSPERSON"]->* (dst IS "PERSON")
  WHERE src."FIRSTNAME" = 'Scott'
  COLUMNS (
    VERTEX_ID(dst) AS person_id,
    dst."FIRSTNAME" AS person_firstname,
    dst."LASTNAME" AS person_lastname
  )
)
```

Intermittent Queries

LLMs have been shown to struggle when translating queries that require more than one GRAPH_TABLE operator. The following are such examples:

Prompt: Show people who have liked all the same posts as Hans

```
SELECT person_id, person_name
FROM GRAPH_TABLE("PERSON_GRAPH"
  MATCH (hans IS "PERSON") -[likes_hans IS "PERSONLIKESPOST"]-> (post IS
"POST"),
  (other_person IS "PERSON") -[likes_other IS "PERSONLIKESPOST"]-> (post)
  WHERE hans."FIRSTNAME" = 'Hans'
  COLUMNS (VERTEX_ID(other_person) as person_id, other_person."FIRSTNAME" AS
person_name)
)
WHERE NOT EXISTS (
  SELECT 1
  FROM GRAPH_TABLE("PERSONGRAPH"
    MATCH (hans IS "PERSON") -[likes_hans IS "PERSONLIKESPOST"]-> (post IS
"POST")
    WHERE hans."FIRSTNAME" = 'Hans'
    COLUMNS (VERTEX_ID(post) as post_id)
  ) hans_posts
  LEFT JOIN GRAPH_TABLE("PERSONGRAPH"
    MATCH (other_person IS "PERSON") -[likes_other IS "PERSONLIKESPOST"]->
(post
is "POST")
    COLUMNS (VERTEX_ID(post) as post_id)
  ) other_posts
  ON hans_posts.post_id = other_posts.post_id
  WHERE other_posts.post_id IS NULL
)
```

Example: Matching of Recursive Path Patterns with Defined Bounds.

Prompt: Find all names of the people that can be reached in a 1 to 3 edge path

```
SELECT person_name
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
  MATCH (start_person IS "PERSON") -[e IS "KNOWS"]->{1,3} (end_person IS
"PERSON")
  COLUMNS (end_person."FIRSTNAME" AS person_name))
```

Example: Filtering Data for Nodes Along a Recursive Path

Prompt: Find all names of the people that can be reached in a 1 to 3 edge path where each person is younger than the next one

```
SELECT person_name
FROM GRAPH_TABLE("ADB_USER"."PERSON_GRAPH"
    MATCH (start_person IS "PERSON") ((v1 IS "PERSON") -[e IS "KNOWS"]-> (v2
IS"PERSON") WHERE v1."AGE" < v2."AGE"){1,3} (end_person IS "PERSON")
    COLUMNS (end_person."FIRSTNAME" AS person_name))
```

Example: Grouping and Aggregation

LLMs often struggle with translating queries that require grouping and aggregation. A common mistake is placing aggregations in the COLUMNS clause instead of the SELECT clause.

Prompt: Find the average number of posts liked by all the users

```
SELECT AVG(COUNT(post)) AS average_liked_count
FROM GRAPH_TABLE("PERSON_GRAPH"
    MATCH (p IS "PERSON") -[e IS "PERSONLIKESPOST"]-> (post IS "POST")
    COLUMNS (VERTEX_ID(p) AS person, VERTEX_ID(post) AS post))
GROUP BY person;
```

Example: Improve SQL Query Generation

These examples demonstrate how comments, annotations, foreign key, and referential integrity constraints in database tables and columns can improve the generation of SQL queries from natural language prompts.

Example: Improve SQL Generation with Table and Column Comments

If you have table and column comments in your database tables, enable "comments":"true" parameter in DBMS_CLOUD_AI.CREATE_PROFILE function to retrieve table level and column level comments. The comments are added to the metadata of the LLM for a better SQL generation.

```
-- Adding comments to table 1, table 2, and table 3. Table 1 has 3 columns,
table 2 has 7 columns, table 3 has 2 columns.
```

```
-- TABLE1
COMMENT ON TABLE table1 IS 'Contains movies, movie titles and the year it was
released';
COMMENT ON COLUMN table1.c1 IS 'movie ids. Use this column to join to other
tables';
COMMENT ON COLUMN table1.c2 IS 'movie titles';
COMMENT ON COLUMN table1.c3 IS 'year the movie was released';
-- TABLE2
COMMENT ON TABLE table2 IS 'transactions for movie views - also known as
streams';
COMMENT ON COLUMN table2.c1 IS 'day the movie was streamed';
COMMENT ON COLUMN table2.c2 IS 'genre ids. Use this column to join to other
tables';
COMMENT ON COLUMN table2.c3 IS 'movie ids. Use this column to join to other
tables';
COMMENT ON COLUMN table2.c4 IS 'customer ids. Use this column to join to
other tables';
COMMENT ON COLUMN table2.c5 IS 'device used to stream, watch or view the
movie';
```

```

COMMENT ON COLUMN table2.c6 IS 'sales from the movie';
COMMENT ON COLUMN table2.c7 IS 'number of views, watched, streamed';

-- TABLE3
COMMENT ON TABLE table3 IS 'Contains the genres';
COMMENT ON COLUMN table3.c1 IS 'genre id. use this column to join to other
tables';
COMMENT ON COLUMN table3.c2 IS 'name of the genre';

BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'myprofile',
    attributes =>
      '{"provider": "azure",
        "azure_resource_name": "my_resource",
        "azure_deployment_name": "my_deployment",
        "credential_name": "my_credential",
        "comments": "true",
        "object_list": [
          {"owner": "moviestream", "name": "table1"},
          {"owner": "moviestream", "name": "table2"},
          {"owner": " moviestream", "name": "table3"}
        ]
      }'
  );

  DBMS_CLOUD_AI.SET_PROFILE(
    profile_name => 'myprofile'
  );

END;
/

--Prompts
select ai what are our total views;
RESPONSE
-----
TOTAL_VIEWS
-----
97890562

select ai showsql what are our total views;

RESPONSE
-----
SELECT SUM(QUANTITY_SOLD) AS total_views
FROM "moviestream"."table"

select ai what are our total views broken out by device;

DEVICE                                TOTAL_VIEWS
-----
mac                                    14719238
iphone                                20793516
ipad                                   15890590

```

pc	14715169
galaxy	10587343
pixel	10593551
lenovo	5294239
fire	5296916

8 rows selected.

select ai showsql what are our total views broken out by device;

RESPONSE

```
-----
-----
SELECT DEVICE, COUNT(*) AS TOTAL_VIEWS
FROM "moviestream"."table"
GROUP BY DEVICE
```

Example: Improve SQL Query Generation with Table and Column Annotations

This example demonstrates the integration of annotations in Select AI, applicable in Oracle Database 23ai. The annotations are added to the metadata that is sent to the LLM.

If you have a table with annotations in your schema, enable "annotations": "true" in the DBMS_CLOUD_AI.CREATE_PROFILE function to instruct Select AI to add annotations to the metadata.

```
--
-- Annotations
--

CREATE TABLE emp2 (
    empno NUMBER,
    ename VARCHAR2(50) ANNOTATIONS (display 'lastname'),
    salary NUMBER ANNOTATIONS ("person_salary", "column_hidden"),
    deptno NUMBER ANNOTATIONS (display 'department')
) ANNOTATIONS (requires_audit 'yes', version '1.0', owner 'HR Organization');
```

Table created.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'GOOGLE_ANNOTATIONS',
    attributes   => '{"provider": "google",
                  "credential_name": "GOOGLE_CRED",
                  "object_list": [{"owner": "ADB_USER", "name": "emp2"}],
                  "annotations" : "true"
                }');
END;
/
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI.SET_PROFILE('GOOGLE_ANNOTATIONS');
```

PL/SQL procedure successfully completed.

Example: Improve SQL Query Generation with Foreign Key and Referential Key Constraints

This example demonstrates the ability of the LLM to generate accurate JOIN conditions by retrieving the foreign key and referential key constraints into the metadata of the LLM. The foreign key and referential key constraints provide structured relationship data between the tables to the LLM.

Enable "constraints": "true" in the DBMS_CLOUD_AI.CREATE_PROFILE function for Select AI to retrieve foreign key and referential key.

```
--
-- Referential Constraints
--
CREATE TABLE dept_test (
    deptno NUMBER PRIMARY KEY,
    dname VARCHAR2(50)
);

Table created.

CREATE TABLE emp3 (
    empno NUMBER PRIMARY KEY,
    ename VARCHAR2(50),
    salary NUMBER,
    deptno NUMBER,
    CONSTRAINT emp_dept_fk FOREIGN KEY (deptno) REFERENCES dept_test(deptno)
);

Table created.

BEGIN
    DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name=>'GOOGLE_CONSTRAINTS',
        attributes => '{"provider": "google",
            "credential_name": "GOOGLE_CRED",
            "object_list": [{ "owner": "ADB_USER", "name": "dept_test"},
                { "owner": "ADB_USER", "name": "emp3"}],
            "constraints" : "true"
        }');
END;
/

PL/SQL procedure successfully completed.

EXEC DBMS_CLOUD_AI.SET_PROFILE('GOOGLE_CONSTRAINTS');
```

PL/SQL procedure successfully completed.

Example: Automatically Detect Relevant Table Metadata

These examples shows how Select AI automatically detects relevant tables and sends metadata only for those specific tables relevant to the query in Oracle Database 23ai. To enable this feature, set object_list_mode to *automated*. This automatically creates a vector index named <profile_name>_OBJECT_LIST_VECINDEX. The vector index is initialized with default attributes and values such as refresh_rate, similarity_threshold, and match_limit.

You can modify some of the attributes through `DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX`. See [#unique_577](#) for more information.

One profile is configured to use `object_list` to specify the schema or the objects in the schema while the other does not specify `object_list`. However, the same SQL construct is expected.

Review [#unique_578](#) to provide access to the `DBMS_CLOUD_AI` package and provide network access to the AI provider.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name=>'OCI_AUTO',
    attributes=> '{"provider":
"oci",
  "credential_name": "GENAI_CRED",
  "object_list": [{"owner": "SH"}],
  "oci_compartment_id": "ocid1.compartment.oc1..aaaa...",
  "model" : "meta.llama-3.3-70b-instruct"
}');
END;
/
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_AUTO');
```

PL/SQL procedure successfully completed.

```
select ai showsql how many customers in San Francisco are married;
```

RESPONSE

```
-----
-----
-----
SELECT COUNT(DISTINCT c."CUST_ID") AS "NUMBER_OF_CUSTOMERS"
FROM "SH"."CUSTOMERS" c
WHERE UPPER(c."CUST_CITY") = UPPER('San Francisco')
AND UPPER(c."CUST_MARITAL_STATUS") = UPPER('married')
```

The following example compares the same scenario without using `object_list`. When you don't specify `object_list`, Select AI automatically chooses all objects available to the current schema.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name=>'OCI_AUTO1',
    attributes=> '{"provider":
"oci",
  "credential_name": "GENAI_CRED",
  "oci_compartment_id": "ocid1.compartment.oc1..aaaa...",
  "object_list_mode": "automated",
  "model" : "meta.llama-3.3-70b-instruct"
}');
END;
/
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_AUTO1');
```

PL/SQL procedure successfully completed.

```
select ai showsql how many customers in San Francisco are married?;
```

RESPONSE

```
-----
-----
-----
SELECT COUNT(c."CUST_ID") AS "Number_of_Customers"
FROM "SH"."CUSTOMERS" c
WHERE UPPER(c."CUST_CITY") = UPPER('San Francisco')
AND UPPER(c."CUST_MARITAL_STATUS") = UPPER('Married')
```

Example: Select AI Summarize

These examples show how to use the summarize action and `DBMS_CLOUD_AI.SUMMARIZE` function. Also, customize the summary generation for your content using the function.

Example: Use Summarize Action on SQL Command Line

The following example uses `SUMMARIZE` as a Select AI action. Use `SELECT AI SUMMARIZE <TEXT>` in the SQL command line to generate a summary of input text.

```
SELECT AI SUMMARIZE
```

```
Like countless other people around the globe, I stream music, and like more
than six hundred million of them I mainly use Spotify. Streaming currently
accounts for about eighty per cent of the American recording industry's
revenue, and in recent years Spotify's health is often consulted as a measure
for the health of the music business over all. Last spring, the International
Federation of the Phonographic Industry reported global revenues of $28.6
billion, making for the ninth straight year of growth. All of this was
unimaginable in the two-thousands, when the major record labels appeared
poorly equipped to deal with piracy and the so-called death of physical media.
On the consumer side, the story looks even rosier. Adjusted for inflation, a
```

```
... (skipped 1000 rows in the middle)
```

```
Pelly writes of some artists, in search of viral fame, who surreptitiously use
social media to effectively beta test melodies and motifs, basically putting
together songs via crowdsourcing. Artists have always fretted about the
pressure to conform, but the data-driven, music-as-content era feels
different. "You are a Spotify employee at that point," Daniel Lopatin, who
makes abstract electronic music as Oneohtrix Point Never, told Pelly. "If your
art practice is so ingrained in the brutal reality that Spotify has outlined
for all of us, then what is the music that you're not making? What does the
music you're not making sound like?" Listeners might wonder something similar.
What does the music we're not hearing sound like?;
```

RESPONSE

```
-----
The music streaming industry, led by Spotify, has revolutionized the way
```

people consume music, with streaming accounting for 80% of the American recording industry's revenue. However, this shift has also complicated the lives of artists, who struggle to survive in a hyper-abundant present where music is often valued for its convenience rather than its artistic merit. Spotify's algorithms prioritize popularity and profitability over artistic diversity, leading to a homogenization of music and a devaluation of the labor that goes into creating it. Meanwhile, the company's executives reap enormous profits, with CEO Daniel Ek's net worth rivaling that of the wealthiest musicians. As music critic Liz Pelly argues, the streaming economy raises important questions about autonomy, creativity, and the value of art in a world where everything is readily available and easily accessible.

✓ **Tip**

In SQL*Plus, a single quotation mark (') is treated as a string delimiter. If your text contains single quotes, either escape the quote by doubling it (' to '''), or enclose the text using the q'[]' quoting mechanism. If your text contains empty double quotes (""), enclose the text using q'[]' mechanism. For example:

```
SELECT AI SUMMARIZE q'[this's a text]';
```

Example: Use DBMS_CLOUD_AI.SUMMARIZE Procedure to Generate a Summary

These examples demonstrate generating a summary by using different parameters from the DBMS_CLOUD_AI.SUMMARIZE procedure.

You can generate a summary from 3000+ word text stored in an OCI object storage by specifying the object storage link as the `location_uri` parameter and your cloud account credentials as `credential_name` using the DBMS_CLOUD_AI.SUMMARIZE

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
    location_uri => 'https://objectstorage.ca-
toronto-1.oraclecloud.com/n/' ||
    'namespace-string/b/bucketname/o/data_folder/' ||
    'summary/test_4000_words.txt',
    credential_name => 'STORE_CRED',
    profile_name => 'GENAI')
from DUAL;
```

Another way to generate a summary from a text that is stored in an OCI object storage is by using the `content` parameter to call the DBMS_CLOUD.GET_OBJECT procedure.

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
    content => TO_CLOB(
        DBMS_CLOUD.GET_OBJECT(
            credential_name => 'STORE_CRED',
            location_uri => 'https://objectstorage.ca-
toronto-1.oraclecloud.com/n/' ||
            'namespace-string/b/bucketname/o/data_folder/' ||
            'summary/test_4000_words.txt')),
    profile_name => 'GENAI')>
from DUAL;
```


Example: Generate a Summary by Specifying User Prompt, Minimum Words, and Maximum Words

The following example demonstrates generating a summary of a 3000+ word text by specifying the following parameters:

- `user_prompt`: *The summary should start with 'The summary of the article is: '*
- `min_words`: 50
- `max_words`: 100

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
    content => TO_CLOB(
        DBMS_CLOUD.GET_OBJECT(
            credential_name => 'STORE_CRED',
            location_uri => 'https://objectstorage.ca-
toronto-1.oraclecloud.com/n/' ||
                                'namespace-string/b/bucketname/o/
data_folder/' ||
                                'summary/test_4000_words.txt')),
    profile_name => 'GENAI',
    user_prompt => 'The summary should start with ''The
summary of ' ||
                                'the article is: ''',
    params => '{"min_words":50,"max_words":100}')
As response FROM dual;
```

RESPONSE

```
-----
--
The summary of the article is: The music streaming industry, led by Spotify,
has
    revolutionized the way people consume music, with streaming accounting for
about
    t eighty per cent of the American recording industry's revenue. However, this
sh
    ift has also raised concerns about the impact on artists, with many
struggling t
    o make a living due to low royalty rates and the dominance of playlists. The
art
    icle explores the history of music streaming, from the early days of Napster
to
    the current landscape, and how it has changed the way people listen to music.
It
    also delves into the issues of autonomy and creativity in the music
industry, w
    ith some artists feeling pressured to conform to certain styles or formulas
to s
    ucced on platforms like Spotify. The article cites examples of artists who
have
    spoken out against the streaming economy, including Taylor Swift and Neil
Young
    , and discusses the rise of alternative platforms like Bandcamp and Nina.
Ultima
    tely, the article suggests that the streaming economy has created a perverse
vis
```

ion for art, where music is valued for its ability to be ignored rather than app
reciated, and that this has significant implications for the future of music
and
creativity. With the rise of AI-generated music and the increasing
importance o
f data-driven decision making in the music industry, the article asks what
the m
usic we're not hearing sounds like, and what the consequences of this shift
will
be for artists and listeners alike. The article concludes by highlighting
the n
eed for a more nuanced understanding of the music industry and the impact of
str
eamming on artists and listeners, and for alternative models that prioritize
crea
tivity and autonomy over profit and convenience.

Example: Generate a Summary by Specifying User Prompt, Maximum Words, and Summary Style

The following example demonstrates generating a summary of a 12000+ word text by specifying the following parameters:

- `user_prompt`: *The summary should start with 'The summary of the article is: '*
- `max_words`: *100*
- `summary_style`: *list*

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
    location_uri      => 'https://objectstorage.ca-toronto-1.' ||
                        'oraclecloud.com/n/namespace-string/b/' ||
                        '/bucketname/o/data_folder/' ||
                        'summary/dreams.txt',
    credential_name   => 'STORE_CRED',
    profile_name      => 'GENAI',
    user_prompt       => 'The summary should start with ''The
summary of ' ||
                        'the article is: ''',
    params            => '{"max_words":100,
"summary_style":"list"}')
As response FROM dual;
```

RESPONSE

```
-----
--
The summary of the article is:
- The book "Dreams" by Henri Bergson explores the concept of dreams and their si
gnificance in understanding human consciousness.
- Bergson argues that dreams are not just random thoughts, but rather a way
for
our unconscious mind to process and consolidate memories.
- He suggests that dreams are a result of the relaxation of our mental
faculties
, which allows our unconscious mind to freely associate and create new
```

connections between memories.

- The book also discusses the role of sensations, such as visual and auditory impressions, in shaping our dreams.
- Bergson's theory of dreams is compared to other theories, including those of Freud and Jung, and is seen as a unique and insightful contribution to the field of psychology.
- The book concludes by highlighting the importance of studying dreams in order to gain a deeper understanding of human consciousness and the workings of the mind.

Example: Generate a Summary of a Book

This example demonstrates passing a 35.66 MiB file as an input to generate a summary. The `DBMS_CLOUD_AI.SUMMARIZE` function uses iterative refinement method to process the chunks. See [#unique_579](#) for more information.

```
SELECT DBMS_CLOUD_AI.SUMMARIZE(
    location_uri => 'https://objectstorage.ca-
toronto-1.oraclecloud.com/n/namespace-string/b/' ||
    'bucketname/o/data_folder/summary/
Descartes_An_Intellectual_Biography.pdf',
    credential_name => 'STORE_CRED',
    profile_name => 'GENAI',
    params =>
    '{"chunk_processing_method":"iterative_refinement"}')
AS response FROM dual;
```

RESPONSE

```
-----
--
Stephen Gaukroger's intellectual biography of Rene Descartes provides a
detailed
examination of the philosopher's crucial role in shaping modern thought,
placing
him within the cultural, religious, and scientific context of the early
seventeenth century. It traces Descartes' intellectual journey from his education
at Le
Fleche, where he rejected Aristotelian logic, to his influential
interactions
with figures like Isaac Beeckman, which shaped his mechanistic worldview
evident
in works like his hydrostatics manuscript and *Compendium Musicae*. The
biography
underscores Descartes' dual commitment to philosophy and science,
highlighting
his social status among the gentry, mathematical innovations such as
solving the
Pappus problem through algebraic geometry, and his epistemology based on
```

clear and distinct ideas. It explores his mechanistic explanations of bodily functions, challenging traditional soul-body distinctions, and his extensive natural philosophy in texts like *Le Monde* and *L'Homme*. Gaukroger also delves into Descartes' cosmological theories, including the vortex theory and laws of motion linked to divine immutability, as well as his nuanced perspectives on animal cognition versus human consciousness. Central to the narrative is Descartes' use of hyperbolic doubt to combat skepticism and establish metaphysical foundations through the *cogito*, alongside his classification of ideas and theological proofs of God's existence. The complex relationship between his natural philosophy and metaphysics, especially in defining motion as a mode, and his innovative approach to the passions in *Passions of the Soul*, rejecting Stoic views for a mind-body union, are key themes. This portrayal captures Descartes' struggle with traditional paradigms during a transformative era, emphasizing his enduring impact on philosophy and science.

Example: Select AI Translate

These examples demonstrate how you can use the translate capability.

To use the Select AI translation feature, you must have the appropriate IAM policy permissions to access Oracle Cloud Infrastructure Language services.

Grant the permission to use `ai-service-language-family` resource in your IAM policy. An example policy statement to grant permission to a user group in a specific compartment is:

```
allow group <your group name> to use ai-service-language-family in
compartment <your_compartment>
```

- If using Resource Principal credential, assign the permission to the Dynamic Group.
- If using Private Key credential, assign the permission to the User Group.

A Dynamic Group identifies resources such as databases or functions by matching their OCIDs or tags, while a User Group contains individual IAM users.

Use a dynamic group when the policy applies to OCI resources, and use a user group when the policy applies to human users. For detailed steps to create dynamic and user groups, see [Managing Dynamic Groups](#).

See [Language Policies](#) for more information.

Example: Use Translate Action on the SQL Command Line

The following example shows using the translate action on the SQL command line.

Note

Your AI profile must specify the target language. This capability is supported only for the provider OCI.

```
--Create an AI profile with language parameters
BEGIN
DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name
=>'GENAI_NEW',
    attributes => '{"provider":
"oci",
    "credential_name": "GENAI_CRED",
    "target_language": "french",
    "object_list": [{"owner": "SH", "name": "customers"},
                    {"owner": "SH", "name": "countries"},
                    {"owner": "SH", "name": "supplementary_demographics"},
                    {"owner": "SH", "name": "profits"},
                    {"owner": "SH", "name": "promotions"},
                    {"owner": "SH", "name": "products"}]
    }');
END;
/
PL/SQL procedure successfully completed.

SQL> exec DBMS_CLOUD_AI.SET_PROFILE('GENAI_NEW');

PL/SQL procedure successfully completed.

SQL> select ai translate I need to translate this;

RESPONSE
-----
Je dois traduire ceci
```

Example: Use Translate in DBMS_CLOUD_AI.GENERATE Function

The following examples show using translate as a Select AI action within the DBMS_CLOUD_AI.GENERATE function. See [#unique_580](#) for more information.

Note

The AI profile can skip specifying the target language parameter if it is passed as an attribute in DBMS_CLOUD_AI.GENERATE.

The translate action is supplied in the DBMS_CLOUD_AI.GENERATE function along with target_language and source_language. This example uses generative AI translation. The input text this is a document in English (source_language: "en") is translated into French (target_language: "fr").

```
SELECT DBMS_CLOUD_AI.GENERATE('select ai translate text to be translated')
      FROM dual;
```

```

DECLARE
    l_attributes clob := '{"target_language": "fr", "source_language":
"en"}';
    output clob;
BEGIN
    output := DBMS_CLOUD_AI.GENERATE(
        prompt          => 'this is a document',
        profile_name     => 'oci_translate',
        action           => 'translate',
        attributes       => l_attributes
    );

```

Example: Use DBMS_CLOUD_AI.TRANSLATE Function for Translation

This example calls the DBMS_CLOUD_AI.TRANSLATE function to use generative AI translation, converting the input text from English (source_language) into French (target_language) using the specified AI profile.

See [#unique_581](#) for more details.

```

BEGIN
    output_text := DBMS_CLOUD_AI.TRANSLATE(
        profile_name => 'GENAI_NEW'
        text         => 'text to be translated',
        source_language => 'English',
        target_language => 'French');
END;
/

```

Example: Display Supported Languages for a Provider

Query the AI_TRANSLATION_LANGUAGES view to see a list of languages that your AI provider supports. See [#unique_582](#) for details.

```
SELECT* FROM AI_TRANSLATION_LANGUAGES;
```

LANGUAGE_NAME	LANGUAGE_CODE	PROVIDER
-----	-----	-----
ARABIC	ar	OCI
CROATIAN	hr	OCI
CZECH	cs	OCI
DANISH	da	OCI
GERMAN	de	OCI
GREEK	el	OCI
ENGLISH	en	OCI
SPANISH	es	OCI
FINNISH	fi	OCI
FRENCH	fr	OCI
FRENCH CANADA	fr-CA	OCI
HEBREW	he	OCI
HUNGARIAN	hu	OCI
ITALIAN	it	OCI

Example: Use a Different Profile for Natural Language to SQL Generation

To use a different profile for natural language to SQL generation, see [#unique_583](#).

Note

The profile in the provided example does not support translate capability.

Select AI with Retrieval Augmented Generation (RAG)

Select AI with RAG augments your natural language prompt by retrieving content from your specified vector store using semantic similarity search. This reduces hallucinations by using your specific and up-to-date content and provides more relevant natural language responses to your prompts.

Select AI automates the Retrieval Augmented Generation (RAG) process. This technique retrieves data from enterprise sources using AI vector search and augments user prompts for your specified large language model (LLM). By leveraging information from enterprise data stores, RAG reduces hallucinations and generates grounded responses.

RAG uses AI vector search on a vector index to find semantically similar data for the specified question. Vector store processes vector embeddings, which are mathematical representations of various data points like text, images, and audio. These embeddings capture the meaning of the data, enabling efficient processing and analysis. For more details on vector embeddings and AI vector search, see Overview of AI Vector Search.

Select AI integrates with AI vector search available in Oracle Autonomous AI Database 26ai for similarity search using vector embeddings.

Benefits of Select AI RAG

Simplify querying, enhance response accuracy with current data, and gain transparency by reviewing sources used by the LLM.

Select AI RAG offers the following benefits:

- **Simplify data querying and increase response accuracy:** Enable users to query enterprise data using natural language and provide LLMs with detailed context from enterprise data to generate more accurate and relevant responses, reducing instances of LLM hallucinations.
- **Up-to-date information:** Provide LLMs access to current enterprise information using vector stores, eliminating the need for costly, time-consuming fine-tuning of LLMs trained on static data sets.
- **Seamless integration:** Integrate with Oracle AI Vector Search for streamlined data handling and enhanced performance.
- **Automated data orchestration:** Automate orchestration steps with a fully managed Vector Index pipeline, ensuring efficient processing of new data.
- **Understandable contextual results:** Has access and retrieves the sources used by the LLM from vector stores, ensuring transparency and confidence in results. Views and extracts data in natural language text or JSON format for easier integration and application development.

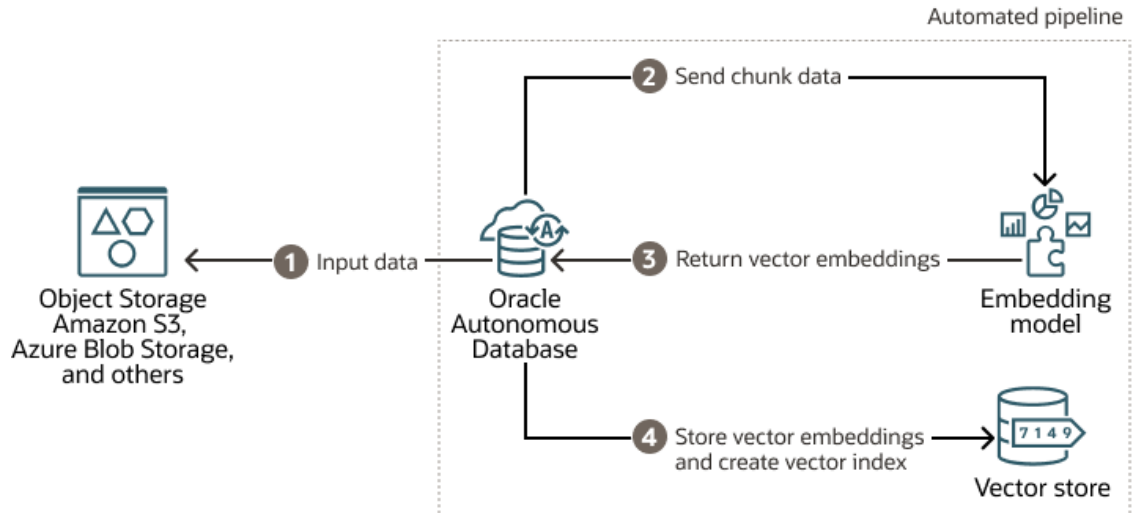
Build your Vector Store

Select AI automates the creation and population of vector store by converting input documents (for example, PDF, DOC, JSON, XML, or HTML) from your object store to plain text. Oracle

Text supports around 150 file types. For a complete list of all the supported document formats, see Supported Document Formats.

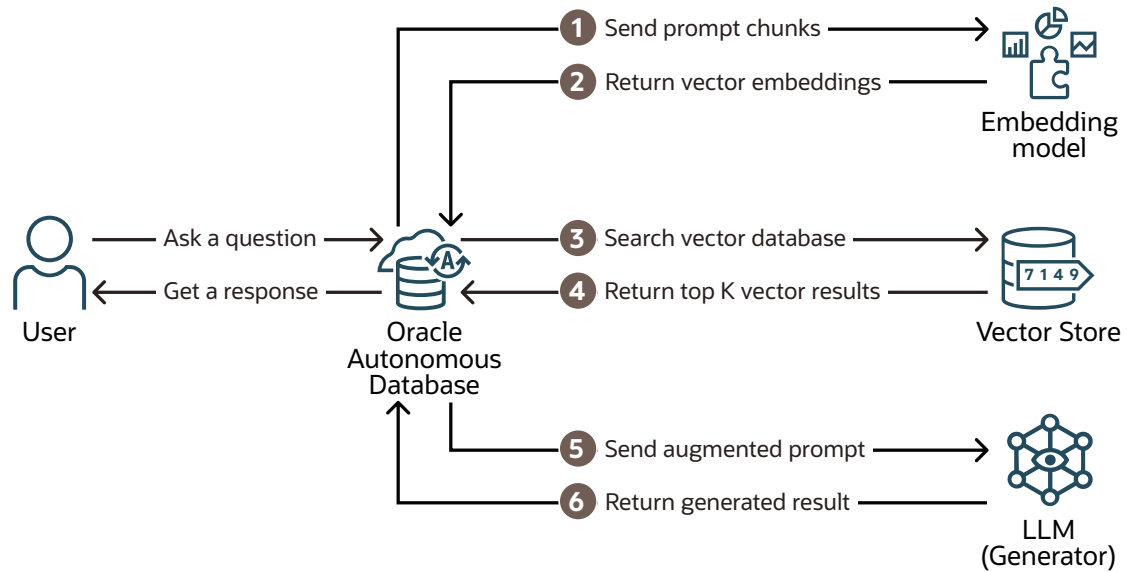
Select AI automatically processes documents to chunks, generates embeddings, stores them in the specified vector store, and updates the vector index as new data arrives.

Here is how the input from Object Storage is used with Select AI RAG:



1. Input: Data is initially stored in an Object Storage.
2. Oracle Autonomous Database retrieves the input data or the document, chunks it and sends the chunks to an embedding model.
3. The embedding model processes the chunk data and returns vector embeddings.
4. The vector embeddings are then stored in a vector store for use with RAG. As content is added, the vector index is automatically updated.

RAG retrieves relevant pieces of information from the enterprise database to answer a user's question. This information is provided to the specified large language model along with the user prompt. Select AI uses this additional enterprise information to enhance the prompt, improving the LLM's response. RAG can enhance response quality with update-to-date enterprise information from the vector store.



Select AI implements RAG as follows:

1. Input: User asks a question (specifies a prompt) using Select AI narrate action.
2. Select AI generates vector embeddings of the prompt using the embedding model specified in the AI profile.
3. The vector search index uses the vector embedding of the question to find matching content from the customer's enterprise data (searching the vector store) which has been indexed.
4. The vector search returns top K texts similar to the input to your Autonomous AI Database instance.
5. Autonomous AI Database then sends these top K query results with user question to the LLM.
6. The LLM returns its response to your Autonomous AI Database instance.
7. Autonomous AI Database Select AI provides the response to the user.

Use DBMS_CLOUD_AI to Create and Manage Vector Indexes

Use the DBMS_CLOUD_AI package to create and manage vector indexes and configure vector database JSON parameters.

After you create credentials and provide network access to the vector database and the AI provider, your Autonomous AI Database instance uses AI profiles to configure access to LLMs. See Example: Set Up and Use Select AI with RAG for a complete example on setting it up and using it in Select AI statements.

Note

If you do not want table data or vector search documents to be sent to an LLM, a user with administrator privileges can disable such access for all users of the given database. This, in effect, disables the narrate action for RAG.

You can configure AI profiles for providers listed in [#unique_588](#) through the DBMS_CLOUD_AI package.

① See Also

- Create a vector index: CREATE_VECTOR_INDEX Procedure.
- Manage vector index profiles and other AI profiles: Summary of DBMS_CLOUD_AI Subprograms.
- Query vector index views: DBMS_CLOUD_AI Views.

Use In-database Transformer Models

Select AI RAG enables you to use pretrained ONNX transformer models that are imported into your database in Oracle Database 23ai instance for generating embedding vectors from document chunks and user prompts.

To learn more about importing a pretrained ONNX model into your Oracle Database 23ai instance, see

- Import Pretrained Models in ONNX Format for Vector Generation Within the Database
- Alternate Method to Import ONNX Models
- [Pre-built Embedding Generation model for Oracle Database 26ai](#)

① Note

You must import a pretrained ONNX-format transformer model into Oracle Database 23ai instance to use Select AI RAG with imported in-database transformer model. You can also use other transformer models from supported AI providers.

See Example: Select AI with In-database Transformer Models to explore the feature.

Example: Set Up and Use Select AI with RAG

This example guides you through setting up credentials, configuring network access, and creating a vector index for integrating OCI Generative AI vector store cloud services with OpenAI using Oracle Autonomous AI Database.

The setup concludes with creating an AI profile that uses the vector index to enhance LLM responses. Finally, this example uses the Select AI narrate action, which returns a response that has been enhanced using information from the specified vector database.

The following example demonstrates building and querying vector index in Oracle Database 23ai.

```
--Grants EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--Grants EXECUTE privilege DBMS_CLOUD_PIPELINE to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;

-- Create the OpenAI credential
```

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OPENAI_CRED',
        username => 'OPENAI_CRED',
        password => '<your_api_key>'
    );
END;
/
```

PL/SQL procedure successfully completed.

```
-- Append the OpenAI endpoint
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'api.openai.com',
        ace => xs$ace_type(privilege_list => xs$name_list('http'),
            principal_name => 'ADB_USER',
            principal_type => xs_acl.ptype_db)
    );
END;
/
```

PL/SQL procedure successfully completed.

```
-- Create the object store credential
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OCI_CRED',
        username => '<your_username>',
        password => '<OCI_profile_password>'
    );
END;
/
```

PL/SQL procedure successfully completed.

-- Create the profile with the vector index.

```
BEGIN
    DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name => 'OPENAI_ORACLE',
        attributes => '{"provider": "openai",
            "credential_name": "OPENAI_CRED",
            "vector_index_name": "MY_INDEX",
            "temperature": 0.2,
            "max_tokens": 4096,
            "model": "gpt-3.5-turbo-1106"
        }');
END;
/
```

PL/SQL procedure successfully completed.

```
-- Set profile
EXEC DBMS_CLOUD_AI.SET_PROFILE('OPENAI_ORACLE');
```

PL/SQL procedure successfully completed.

```
-- create a vector index with the vector store name, object store location and
-- object store credential
```

```
BEGIN
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name => 'MY_INDEX',
        attributes => '{"vector_db_provider": "oracle",
                        "location": "https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
                        "object_storage_credential_name": "OCI_CRED",
                        "profile_name": "OPENAI_ORACLE",
                        "vector_dimension": 1536,
                        "vector_distance_metric": "cosine",
                        "chunk_overlap": 128,
                        "chunk_size": 1024
                       }');
END;
/
```

PL/SQL procedure successfully completed.

```
-- After the vector index is populated, we can now query the index.
```

```
-- Set profile
EXEC DBMS_CLOUD_AI.SET_PROFILE('OPENAI_ORACLE');
```

PL/SQL procedure successfully completed.

```
-- Select AI answers the question with the knowledge available in the vector
database.
```

```
set pages 1000
set linesize 150
SELECT AI narrate how can I deploy an oracle machine learning model;
RESPONSE
To deploy an Oracle Machine Learning model, you would first build your model
within the Oracle database. Once your in-database models are built, they
become immediately available for use, for instance, through a SQL query using
the prediction operators built into the SQL language.
```

The model scoring, like model building, occurs directly in the database, eliminating the need for a separate engine or environment within which the model and corresponding algorithm code operate. You can also use models from a different schema (user account) if the appropriate permissions are in place.

Sources:

- Manage-your-models-with-Oracle-Machine-Learning-on-Autonomous-Database.txt (https://objectstorage.../v1/my_namespace/my_bucket/my_data_folder/Manage-your-models-with-Oracle-Machine-Learning-on-Autonomous-Database.txt)

```
- Develop-and-deploy-machine-learning-models-using-Oracle-Autonomous-Database-Machine-Learning-and-APEX.txt (https://objectstorage.../v1/my_namespace/my_bucket/my_data_folder/Develop-and-deploy-machine-learning-models-using-Oracle-Autonomous-Database-Machine-Learning-and-APEX.txt)
```

Example: Select AI with In-database Transformer Models

This example demonstrates how you can import a pretrained transformer model that is stored in Oracle object storage into your Oracle Database 23ai instance and then use the imported in-database model in Select AI profile to generate vector embeddings for document chunks and user prompts.

To use in-database transformer models in your Select AI profile, be sure you have:

- your pretrained model imported in your Oracle Database 23ai instance.
- optionally, access to Oracle object storage.

Import a Pretrained Transformer Model into your Oracle Database 23ai From Oracle Object Storage

Review the steps in Import Pretrained Models in ONNX Format for Vector Generation Within the Database and the blog [Pre-built Embedding Generation model for Oracle AI Database 26ai](#) to import a pretrained transformer model into your database.

The following example shows how to import a pretrained transformer model from Oracle object storage into your database and then view the imported model.

- Create a Directory object, or use an existing directory object

```
CREATE OR REPLACE DIRECTORY ONNX_DIR AS 'onnx_model';
```

-- Object storage bucket

```
VAR location_uri VARCHAR2(4000);
```

```
EXEC :location_uri := 'https://adwc4pm.objectstorage.us-
```

```
ashburn-1.oci.customer-oci.com/p/
```

```
eLddQappgBJ7jNi6Guz9m9L0tYe2u8LWY19GfgU8fLFK4N9YgP4kTlrE9Px3pE12/n/adwc4pm/b/
```

```
OML-Resources/o/';
```

-- Model file name

```
VAR file_name VARCHAR2(512);
```

```
EXEC :file_name := 'all_MiniLM_L12_v2.onnx';
```

-- Download ONNX model from object storage into the directory object

```
BEGIN
```

```
  DBMS_CLOUD.GET_OBJECT(
```

```
    credential_name => NULL,
```

```
    directory_name  => 'ONNX_DIR',
```

```
    object_uri      => :location_uri || :file_name);
```

```
END;
```

```
/
```

-- Load the ONNX model into the database

```
BEGIN
```

```
  DBMS_VECTOR.LOAD_ONNX_MODEL(
```

```
    directory => 'ONNX_DIR',
```

```
    file_name => :file_name,
```

```
    model_name => 'MY_ONNX_MODEL');
```

```

END;
/

-- Verify
SELECT model_name, algorithm, mining_function
FROM user_mining_models
WHERE model_name='MY_ONNX_MODEL';

```

Use In-database Transformer Models in Select AI Profiles

These examples illustrate how to use in-database transformer models within a Select AI profile. One profile is configured only for generating vector embeddings, while the other supports both Select AI actions and vector index creation.

Review Requirements to Configure DBMS_CLOUD_AI Package to complete the prerequisites.

The following is an example for generating vector embeddings only:

```

BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'EMBEDDING_PROFILE',
    attributes   => '{"provider": "database",
                  "embedding_model": "MY_ONNX_MODEL"}'
  );
END;
/

```

The following is an example for general Select AI actions and vector index generation where you can specify a supported AI provider. This example uses OCI Gen AI profile and credentials. See [#unique_588](#) for list of supported providers. However, if you want to use in-database transformer model for generating vector embeddings, then use "database: <MY_ONNX_MODEL>" in embedding_model attribute:

```

BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'GENAI_CRED',
    user_ocid      => 'ocid1.user.oc1..aaaa...',
    tenancy_ocid   => 'ocid1.tenancy.oc1..aaaa...',
    private_key    => '<your_api_key>',
    fingerprint    => '<your_fingerprint>'
  );
END;
/

BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'OCI_GENAI',
    attributes   => '{"provider": "oci",
                  "model": "meta.llama-3.3-70b-instruct",
                  "credential_name": "GENAI_CRED",
                  "vector_index_name": "MY_INDEX",
                  "embedding_model": "database: MY_ONNX_MODEL"}'
  );
END;
/

```

Use Select AI with an In-database Transformer Model from Another Schema

This example demonstrates how to use Select AI with an in-database transformer model if another schema owner owns the model. Specify `schema_name.object_name` as the fully qualified name of the model in `embedding_model` attribute. If the current user is the schema owner or owns the model, you can omit the schema name.

Be sure to have the following privileges if a different schema owner owns the model:

- `CREATE ANY MINING MODEL` system privilege
- `SELECT ANY MINING MODEL` system privilege
- `SELECT MINING MODEL` object privilege on the specific model

To grant a system privilege, you must either have been granted the system privilege with the `ADMIN OPTION` or have been granted the `GRANT ANY PRIVILEGE` system privilege.

See System Privileges for Oracle Machine Learning for SQL to review the privileges.

The following statements allow `ADB_USER1` to score data and view model details in any schema as long as `SELECT` access has been granted to the data. However, `ADB_USER1` can only create models in the `ADB_USER1` schema.

```
GRANT CREATE MINING MODEL TO ADB_USER1;
GRANT SELECT ANY MINING MODEL TO ADB_USER1;
```

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'OCI_GENAI',
    attributes   => '{"provider": "oci",
                    "credential_name": "GENAI_CRED",
                    "vector_index_name": "MY_INDEX",
                    "embedding_model": "database:
ADB_USER1.MY_ONNX_MODEL"}'
  );
END;
/
```

The following example shows how you can specify case sensitive model object name:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'OCI_GENAI',
    attributes   => '{"provider": "oci",
                    "credential_name": "GENAI_CRED",
                    "model": "meta.llama-3.3-70b-instruct",
                    "vector_index_name": "MY_INDEX",
                    "embedding_model": "database:
\"adb_user1\".\"my_model\""}'
  );
END;
/
```

End-to-end Examples with Different AI Providers

These examples demonstrate end-to-end steps for using in-database transformer model with Select AI RAG. One profile uses *database* as the provider exclusively created for generating

embedding vectors while the other profile uses *oci* as the provider created for Select AI actions as well as vector index.

Review Requirements to Configure DBMS_CLOUD_AI Package to complete the prerequisites.

```
--Grant create any directory privilege to the user
GRANT CREATE ANY DIRECTORY to ADB_USER;

- Create a Directory object, or use an existing directory object
CREATE OR REPLACE DIRECTORY ONNX_DIR AS 'onnx_model';

-- Object storage bucket
VAR location_uri VARCHAR2(4000);
EXEC :location_uri := 'https://adwc4pm.objectstorage.us-
ashburn-1.oci.customer-oci.com/p/
eLddQappgBJ7jNi6Guz9m9LOtYe2u8LWY19GfgU8fLFK4N9YgP4kTlrE9Px3pE12/n/adwc4pm/b/
OML-Resources/o/';

-- Model file name
VAR file_name VARCHAR2(512);
EXEC :file_name := 'all_MiniLM_L12_v2.onnx';

-- Download ONNX model from object storage into the directory object
BEGIN
    DBMS_CLOUD.GET_OBJECT(
        credential_name => NULL,
        directory_name   => 'ONNX_DIR',
        object_uri        => :location_uri || :file_name);
END;
/

-- Load the ONNX model into the database
BEGIN
    DBMS_VECTOR.LOAD_ONNX_MODEL(
        directory   => 'ONNX_DIR',
        file_name   => :file_name,
        model_name  => 'MY_ONNX_MODEL');
END;
/

-- Verify
SELECT model_name, algorithm, mining_function
FROM user_mining_models
WHERE model_name='MY_ONNX_MODEL';

--Administrator grants EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--Administrator grants EXECUTE privilege DBMS_CLOUD_PIPELINE to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;

-- Create the object store credential
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OCI_CRED',
```



```

        username => '<your_username>',
        password => '<OCI_profile_password>'
    );
END;
/

```

PL/SQL procedure successfully completed.

-- Create the profile with Oracle Database.

```

BEGIN
    DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name => 'EMBEDDING_PROFILE',
        attributes   => '{"provider": "database",
                        "embedding_model": "MY_ONNX_MODEL"
                        }');
END;
/

```

PL/SQL procedure successfully completed.

```

-- Set profile
EXEC DBMS_CLOUD_AI.SET_PROFILE('EMBEDDING_PROFILE');

```

PL/SQL procedure successfully completed.

This example uses *oci* as the provider.

```

--Grant create any directory privilege to the user
GRANT CREATE ANY DIRECTORY to ADB_USER;

```

```

- Create a Directory object, or use an existing directory object
CREATE OR REPLACE DIRECTORY ONNX_DIR AS 'onnx_model';

```

```

-- Object storage bucket
VAR location_uri VARCHAR2(4000);
EXEC :location_uri := 'https://adwc4pm.objectstorage.us-
ashburn-1.oci.customer-oci.com/p/
eLddQappgBJ7jNi6Guz9m9LOtYe2u8LWY19GfgU8fLFK4N9YgP4kTlrE9Px3pE12/n/adwc4pm/b/
OML-Resources/o/';

```

```

-- Model file name
VAR file_name VARCHAR2(512);
EXEC :file_name := 'all_MiniLM_L12_v2.onnx';

```

```

-- Download ONNX model from object storage into the directory object
BEGIN
    DBMS_CLOUD.GET_OBJECT(
        credential_name => NULL,
        directory_name  => 'ONNX_DIR',
        object_uri       => :location_uri || :file_name);
END;
/

```

```

-- Load the ONNX model into the database
BEGIN
    DBMS_VECTOR.LOAD_ONNX_MODEL(
        directory => 'ONNX_DIR',
        file_name => :file_name,
        model_name => 'MY_ONNX_MODEL');
END;
/

-- Verify
SELECT model_name, algorithm, mining_function
FROM user_mining_models
WHERE model_name='MY_ONNX_MODEL';

--Administrator Grants EXECUTE privilege to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

--Administrator Grants EXECUTE privilege DBMS_CLOUD_PIPELINE to ADB_USER
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;

-- Create the object store credential
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OCI_CRED',
        username => '<your_username>',
        password => '<OCI_profile_password>'
    );
END;
/

--Create GenAI credentials
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'GENAI_CRED',
        user_ocid      => 'ocid1.user.oc1..aaaa...',
        tenancy_ocid   => 'ocid1.tenancy.oc1..aaaa...',
        private_key    => '<your_api_key>',
        fingerprint    => '<your_fingerprint>'
    );
END;
/

--Create OCI AI profile
BEGIN
    DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name => 'OCI_GENAI',
        attributes  => '{"provider": "oci",
                        "model": "meta.llama-3.3-70b-instruct",
                        "credential_name": "GENAI_CRED",
                        "vector_index_name": "MY_INDEX",
                        "embedding_model": "database: MY_ONNX_MODEL"}'
    );
END;
/

-- Set profile

```

```
EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_GENAI');
```

PL/SQL procedure successfully completed.

```
-- create a vector index with the vector store name, object store location and
-- object store credential
```

```
BEGIN
```

```
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name => 'MY_INDEX',
        attributes => '{"vector_db_provider": "oracle",
                        "location": "https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
                        "object_storage_credential_name": "OCI_CRED",
                        "profile_name": "OCI_GENAI",
                        "vector_dimension": 384,
                        "vector_distance_metric": "cosine",
                        "chunk_overlap": 128,
                        "chunk_size": 1024
                    }');
END;
```

```
/
```

PL/SQL procedure successfully completed.

```
-- Set profile
```

```
EXEC DBMS_CLOUD_AI.SET_PROFILE('OCI_GENAI');
```

PL/SQL procedure successfully completed.

```
-- Select AI answers the question with the knowledge available in the vector
database.
```

```
set pages 1000
```

```
set linesize 150
```

```
SELECT AI narrate how can I deploy an oracle machine learning model;
```

```
RESPONSE
```

To deploy an Oracle Machine Learning model, you would first build your model within the Oracle database. Once your in-database models are built, they become immediately available for use, for instance, through a SQL query using the prediction operators built into the SQL language.

The model scoring, like model building, occurs directly in the database, eliminating the need for a separate engine or environment within which the model and corresponding algorithm code operate. You can also use models from a different schema (user account) if the appropriate permissions are in place.

Sources:

- Manage-your-models-with-Oracle-Machine-Learning-on-Autonomous-Database.txt (https://objectstorage.../v1/my_namespace/my_bucket/my_data_folder/Manage-your-models-with-Oracle-Machine-Learning-on-Autonomous-Database.txt)
- Develop-and-deploy-machine-learning-models-using-Oracle-Autonomous-

Database-Machine-Learning-and-APEX.txt
(https://objectstorage.../v1/my_namespace/my_bucket/my_data_folder/Develop-and-deploy-machine-learning-models-using-Oracle-Autonomous-Database-Machine-Learning-and-APEX.txt)

Synthetic Data Generation

Generate synthetic data using random generators, algorithms, statistical models, and Large Language Models (LLMs) to simulate real data for developing and testing solutions effectively.

Note

Support for Synthetic Data Generation is available in Oracle Database 19c starting with version 19.29, and in Oracle AI Database 26ai starting with version 23.26.

Synthetic data can be a powerful tool when developing and testing solutions, especially when actual data doesn't yet exist or isn't allowed to be used. Synthetic, or artificially generated, data can have many of the characteristics of real data. Synthetic data is typically created using random generators, algorithms, or statistical models to simulate the characteristics and distributions of real data. However, this can be complex to produce or rely on tools with features of varying sophistication. With the availability of Large Language Models (LLMs), more relevant and schema-specific data may be generated that considers characteristics expressed in natural language.

Benefits of Synthetic Data Generation

Synthetic data generation enables populating database metadata clones, supporting development, testing, and machine learning projects without using sensitive data from original tables.

Synthetic Data Generation offers the following benefits:

- **Populating metadata clones with synthetic data:** A metadata clone replicates the structure of a database or schema without including actual data. Select AI allows synthetic data generation to populate these clones, protecting sensitive data while enabling development, testing, and creating templates. This approach supports performance and scalability testing.
- **Starting new projects:** When starting a new project, actual data may not be available. Synthetic data provides realistic samples to help demonstrate concepts and gain support for project proposals.
- **Validating user experience:** Synthetic data aids in testing user interfaces by providing diverse data sets to uncover design flaws, performance, and scalability issues.
- **Supporting AI and machine learning projects:** Synthetic data is useful for training AI and machine learning models when real data is unavailable or restricted. LLMs can generate data with specific patterns to facilitate model training and scoring.

Generate Synthetic Data

Use `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA` function to generate synthetic data and query the data using Select AI actions.

To learn more, see `GENERATE_SYNTHETIC_DATA` Function and Example: Generate Synthetic Data.

Monitor and Troubleshoot Synthetic Data Generation

When generating large amounts of data across many tables, Select AI splits synthetic data generation tasks into smaller chunks and runs tasks in parallel. The status of each chunk is tracked in the `SYNTHETIC_DATA$<operation_id>_STATUS` table.

Synthetic data generation operations are logged in the tables `DBA_LOAD_OPERATIONS` and `USER_LOAD_OPERATIONS`. Use these tables to monitor the `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA` operation. See [Track DBMS_CLOUD Load Operations](#) for more details.

After running synthetic data generation in a given session, you can get the latest `<operation_id>` from `USER_LOAD_OPERATION` using the following:

```
SELECT max(id) FROM user_load_operations;
```

To view the synthetic data operations running in a different session, use the `DBA_LOAD_OPERATIONS` view.

View Status of Synthetic Data Generation

The status table of synthetic data generation operation shows the progress of each table and its corresponding chunk. The `STATUS_TABLE` column in `USER_LOAD_OPERATIONS` or `DBA_LOAD_OPERATIONS` shows the status table name. The table name is `SYNTHETIC_DATA$<operation_id>_STATUS` and it has following columns:

Name	Datatype	Description
ID	NUMBER	Unique identifier of the record.
NAME	VARCHAR2	Qualified name of the table, such as "ADB_USER"."EMPLOYEES"
BYTES	NUMBER	Desired number of records for this data generation task
ROWS_LOADED	NUMBER	Actual number of records generated.
CHECKSUM	VARCHAR2	Starting value for the primary key during this data generation task.
LAST_MODIFIED	TIMESTAMP WITH TIME ZONE	Timestamp indicating when the record was last modified.
STATUS	VARCHAR2	Status of the data generation task. The valid values are: • COMPLETED • FAILED • PENDING • SKIPPED
ERROR_CODE	NUMBER	Error code, if the data generation task fails.
ERROR_MESSAGE	VARCHAR2	Error message provided if the task fails.
END_TIME	TIMESTAMP WITH TIME ZONE	Timestamp marking the end of the data generation task.

Example: Check the Number of Records Generated for Each Table

To check the number of records generated for each table, issue the following:

```
SELECT name, SUM(rows_loaded) FROM synthetic_data$<operation_id>_status group
by name;
```

Query ROWS_LOADED to confirm how many number of rows are loaded for each chunk, and SUM(ROWS_LOADED) for rows for each table.

```
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name => 'GENAI',
    object_list => '[{"owner": "ADB_USER", "name":
"DIRECTOR", "record_count":150},
{"owner": "ADB_USER", "name":
"MOVIE_ACTOR", "record_count":300},
{"owner": "ADB_USER", "name": "CLASSES",
"user_prompt":"all in fall semester", "record_count":5},
{"owner": "ADB_USER", "name":
"ACTOR", "record_count":220},
{"owner": "ADB_USER", "name":
"MOVIE", "record_count":50}]'
  );
END;
/

-- Check loaded rows for each chunk
SQL> SELECT name, rows_loaded FROM synthetic_data$141_status order by name;
NAME                                ROWS_LOADED
-----
"ADB_USER"."ACTOR"                  188
"ADB_USER"."ACTOR"                   32
"ADB_USER"."CLASSES"                  5
"ADB_USER"."DIRECTOR"                150
"ADB_USER"."MOVIE"                   50
"ADB_USER"."MOVIE_ACTOR"              38
"ADB_USER"."MOVIE_ACTOR"             114
"ADB_USER"."MOVIE_ACTOR"             148
```

```
-- Check loaded rows for each table
SQL> SELECT name, SUM(rows_loaded) FROM synthetic_data$141_status group by
name;
```

```
NAME                                SUM(ROWS_LOADED)
-----
"ADB_USER"."DIRECTOR"                150
"ADB_USER"."MOVIE_ACTOR"             300
"ADB_USER"."CLASSES"                  5
"ADB_USER"."ACTOR"                   220
"ADB_USER"."MOVIE"                    50
```

Example: Generate Synthetic Data

This example explores how you can generate synthetic data mimicking the characteristics and distribution of real data.

Note

Support for Synthetic Data Generation is available in Oracle Database 19c starting with version 19.29, and in Oracle Database 26ai starting with version 23.26.

The following example shows how to create a few tables in your schema, use OCI Generative AI as your AI provider to create an AI profile, synthesize data into those tables using the `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA` function, and query or generate responses to natural language prompts with Select AI.

--Create tables or use cloned tables

```
CREATE TABLE ADB_USER.Director (
    director_id    INT PRIMARY KEY,
    name           VARCHAR(100)
);
CREATE TABLE ADB_USER.Movie (
    movie_id       INT PRIMARY KEY,
    title           VARCHAR(100),
    release_date    DATE,
    genre           VARCHAR(50),
    director_id     INT,
    FOREIGN KEY (director_id) REFERENCES ADB_USER.Director(director_id)
);
CREATE TABLE ADB_USER.Actor (
    actor_id       INT PRIMARY KEY,
    name           VARCHAR(100)
);
CREATE TABLE ADB_USER.Movie_Actor (
    movie_id       INT,
    actor_id       INT,
    PRIMARY KEY (movie_id, actor_id),
    FOREIGN KEY (movie_id) REFERENCES ADB_USER.Movie(movie_id),
    FOREIGN KEY (actor_id) REFERENCES ADB_USER.Actor(actor_id)
);

-- Create the GenAI credential
BEGIN
    DBMS_CLOUD.create_credential(
        credential_name => 'GENAI_CRED',
        user_ocid       => 'ocid1.user.oc1....',
        tenancy_ocid    => 'ocid1.tenancy.oc1....',
        private_key     => 'vZ6c0...',
        fingerprint     => '86:7d:...'
    );
END;
/
```

```
-- Create a profile
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name
=>'GENAI',
    attributes =>'{"provider":
"oci",
  "credential_name": "GENAI_CRED",
  "object_list": [{"owner": "ADB_USER",
"oci_compartment_id": "ocid1.compartment.oc1...."}]
  }');
END;
/
```

```
EXEC DBMS_CLOUD_AI.set_profile('GENAI');
```

```
-- Run the API for single table
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name => 'GENAI',
    object_name => 'Director',
    owner_name => 'ADB_USER',
    record_count => 5
  );
END;
/
```

PL/SQL procedure successfully completed.

```
-- Query the table to see results
SQL> SELECT * FROM ADB_USER.Director;
```

```
DIRECTOR_ID NAME
```

```
-----
```

```
-----
```

```
1 John Smith
2 Emily Chen
3 Michael Brown
4 Sarah Taylor
5 David Lee
```

```
-- Or ask select ai to show the results
SQL> select ai how many directors are there;
```

```
NUMBER_OF_DIRECTORS
```

```
-----
```

```
5
```

Example: Generate Synthetic Data for Multiple Tables

After you create and set your AI provider profile, use the DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA to generate data for multiple tables. You can query or use Select AI to respond to the natural language prompts.

```
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name => 'GENAI',
    object_list => '[{"owner": "ADB_USER", "name":
"Director", "record_count": 5},
{"owner": "ADB_USER", "name":
"Movie_Actor", "record_count": 5},
{"owner": "ADB_USER", "name":
"Actor", "record_count": 10},
{"owner": "ADB_USER", "name":
"Movie", "record_count": 5, "user_prompt": "all movies released in 2009"}]'
  );
END;
/
PL/SQL procedure successfully completed.
```

```
-- Query the table to see results
SQL> select * from ADB_USER.Movie;
```

MOVIE_ID	TITLE	GENRE	RELEASE_D	DIRECTOR_ID
1	The Dark Knight	Action	15-JUL-09	8
2	Inglourious Basterds	War	21-AUG-09	3
3	Up in the Air	Drama	04-SEP-09	6
4	The Hangover	Comedy	05-JUN-09	1
5	District 9	Science Fiction	14-AUG-09	10

```
-- Or ask select ai to show the results
SQL> select ai how many actors are there;
```

```
Number of Actors
-----
10
```

Example: Guide Synthetic Data Generation with Sample Rows

To guide AI service in generating synthetic data, you can randomly select existing records from a table. For instance, by adding {"sample_rows": 5} to the params argument, you can send 5

sample rows from a table to the AI provider. This example generates 10 additional rows based on the sample rows from the Transactions table.

```
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name => 'GENAI',
    object_name  => 'Transactions',
    owner_name   => 'ADB_USER',
    record_count => 10,
    params       => '{"sample_rows":5}'
  );
END;
/
```

Example: Customize Synthetic Data Generation with User Prompts

The `user_prompt` argument enables you to specify additional rules or requirements for data generation. This can be applied to a single table or as part of the `object_list` argument for multiple tables. For example, in the following calls to `DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA`, the prompt instructs the AI to generate synthetic data on movies released in 2009.

-- Definition for the Movie table CREATE TABLE Movie

```
CREATE TABLE Movie (
  movie_id      INT PRIMARY KEY,
  title         VARCHAR(100),
  release_date  DATE,
  genre         VARCHAR(50),
  director_id   INT,
  FOREIGN KEY (director_id) REFERENCES Director(director_id)
);
```

```
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name  => 'GENAI',
    object_name   => 'Movie',
    owner_name    => 'ADB_USER',
    record_count  => 10,
    user_prompt   => 'all movies are released in 2009',
    params        => '{"sample_rows":5}'
  );
END;
/
```

```
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name => 'GENAI',
    object_list => '[{"owner": "ADB_USER", "name":
"Director", "record_count":5},
                    {"owner": "ADB_USER", "name":
"Movie_Actor", "record_count":5},
                    {"owner": "ADB_USER", "name":
"Actor", "record_count":10}],
```

```

        {"owner": "ADB_USER", "name":
"Movie", "record_count":5, "user_prompt": "all movies are released in 2009"}]}']
    );
END;
/

```

Example: Improve Synthetic Data Quality by Using Table Statistics

If a table has column statistics or is cloned from a database that includes metadata, Select AI can use these statistics to generate data that closely resembles or is consistent with the original data.

For NUMBER columns, the high and low values from the statistics guide the value range. For instance, if the SALARY column in the original EMPLOYEES table ranges from 1000 to 10000, the synthetic data for this column will also fall within this range.

For columns with distinct values, such as a STATE column with values CA, WA, and TX, the synthetic data will use these specific values. You can manage this feature using the `{"table_statistics": true/false}` parameter. By default, the table statistics are enabled.

```

BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name      => 'GENAI',
    object_name       => 'Movie',
    owner_name        => 'ADB_USER',
    record_count      => 10,
    user_prompt       => 'all movies released in 2009',
    params            => '{"sample_rows":5, "table_statistics":true}'
  );
END;
/

```

Example: Use Column Comments to Guide Data Generation

If column comments exist, Select AI automatically includes them to provide additional information for the LLM during data generation. For example, a comment on the Status column in a Transaction table might list allowed values such as *successful*, *failed*, *pending*, *canceled*, and *need manual check*. You can also add comments to further explain the column, giving AI services more precise instructions or hints for generating accurate data. By default, comments are disabled. See Optional Parameters for more details.

```

-- Use comment on column
COMMENT ON COLUMN Transaction.status IS 'the value for state should either be
'successful', 'failed', 'pending' or 'canceled'';
/

BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name      => 'GENAI',
    object_name       => 'employees',
    owner_name        => 'ADB_USER',
    record_count      => 10
    params            => '{"comments":true}'
  );
END;
/

```

Example: Set Unique Values in Synthetic Data Generation

When generating large amounts of synthetic data with LLMs, duplicate values are likely to occur. To prevent this, set up a unique constraint on the relevant column. This ensures that Select AI ignores rows with duplicate values in the LLM response. Additionally, to restrict values for certain columns, you can use the `user_prompt` or add comments to specify the allowed values, such as limiting a `STATE` column to `CA`, `WA`, and `TX`.

```
-- Use 'user_prompt'
BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
        profile_name => 'GENAI',
        object_name  => 'employees',
        owner_name   => 'ADB_USER',
        user_prompt  => 'the value for state should either be CA, WA, or TX',
        record_count => 10
    );
END;
/

-- Use comment on column
COMMENT ON COLUMN EMPLOYEES.state IS 'the value for state should either be
CA, WA, or TX'
/
```

Example: Enhance Synthetic Data Generation by Parallel Processing

To reduce runtime, Select AI splits synthetic data generation tasks into smaller chunks for tables without primary keys or with numeric primary keys. These tasks run in parallel, interacting with the AI provider to generate data more efficiently. The Degree of Parallelism (DOP) in your database, influenced by your Autonomous AI Database on Dedicated Exadata Infrastructure service level and ECPU or OCPU settings, determines the number of records each chunk processes. Running tasks in parallel generally improves performance, especially when generating large amounts of data across many tables. To manage the parallel processing of synthetic data generation, set `priority` as an optional parameter. See [Optional Parameters](#).

Example: Enable or Disable Data Access

This example illustrates how administrators can control data access and prevent Select AI from sending actual schema tables to the LLM.

Disabling Data Access

To restrict access to schema tables, log in as an administrator and run the following procedure.

```
EXEC DBMS_CLOUD_AI.DISABLE_DATA_ACCESS;
```

PL/SQL procedure successfully completed.

Disabling data access limits Select AI's `narrate` action and Synthetic Data Generation. The `narrate` action and synthetic data generation raise an error.

Log in as database user, create and configure your AI profile. Review [Use DBMS_CLOUD_AI to Configure AI Profiles](#) to configure your AI profile.

```
BEGIN
    DBMS_CLOUD_AI.CREATE_PROFILE(
```

```

        profile_name => 'DATA_ACCESS',
        attributes   => '{"provider": "openai",
        "credential_name": "OPENAI_CRED",
        "object_list": [{"owner": "SH"}]
        }');
END;
/

EXEC DBMS_CLOUD_AI.SET_PROFILE('DATA_ACCESS');

select ai how many customers;

NUM_CUSTOMERS
55500

select ai narrate what are the top 3 customers in San Francisco;

ORA-20000: Data access is disabled for SELECT AI.
ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD", line 2228
ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD_AI", line 13157
ORA-06512: at line 1 https://docs.oracle.com/error-help/db/ora-20000/
The stored procedure 'raise_application_error' was called which causes this
error to be generated
Error at Line: 1 Column: 6

```

The following example shows the errors that are triggered when you try to generate synthetic data.

```

BEGIN
DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
profile_name => 'DATA_ACCESS_SDG',
object_name => 'CUSTOMERS_NEW',
owner_name => 'ADB_USER',
record_count => 5
);
END;
/

ERROR at line 1:

ORA-20000: Data access is disabled for SELECT AI.
ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD", line 2228
ORA-06512: at "C##CLOUD$SERVICE.DBMS_CLOUD_AI", line 13401

ORA-06512: at line 2

```

Enabling Data Access

The following example shows enabling data access. Log in as an administrator and run the following procedure:

```

EXEC DBMS_CLOUD_AI.ENABLE_DATA_ACCESS;

PL/SQL procedure successfully completed.

```

Log in as database user, create and configure your AI profile. Review Use DBMS_CLOUD_AI to Configure AI Profiles to configure your AI profile.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'DATA_ACCESS_NEW',
    attributes   => '{"provider": "openai",
                    "credential_name": "OPENAI_CRED",
                    "object_list": [{"owner": "SH"}]
    }');
END;
/
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI.SET_PROFILE('DATA_ACCESS_NEW');
```

PL/SQL procedure successfully completed.

```
select ai how many customers;
```

```
NUM_CUSTOMERS
55500
```

```
select ai narrate what are the top 3 customers in San Francisco;
```

```
"RESPONSE"
"The top 3 customers in San Francisco are Cody Seto, Lauren Yaskovich, and
Ian Mc"
```

The following example shows successful synthetic data generation after enabling data access.

```
BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name => 'DATA_ACCESS_SDG',
    object_name => 'CUSTOMERS_NEW',
    owner_name => 'ADB_USER',
    record_count => 5
  );
END;
/
```

PL/SQL procedure successfully completed.

Select AI Agent

Select AI Agent (autonomous agent framework) enables you to build interactive and autonomous agents inside Autonomous AI Database, combining planning, tool use, reflection, and memory to deliver multi-turn workflows.

Note

Support for Select AI Agent and the DBMS_CLOUD_AI_AGENT Package is available in Oracle Database 19c starting with version 19.29, and in Oracle Database 26ai starting with version 23.26.

About Select AI Agent

Select AI Agent (autonomous agent framework) is a program for creating and managing interactive and autonomous agents inside Autonomous AI Database. Agents reason about requests, call tools, reflect on results, and maintain context with short and long-term memory powered by an AI profile specified LLM with the ReAct (Reasoning and Acting) agentic pattern.

Select AI Agent enables the use of built-in tools such as RAG and Natural Language to SQL (NL2SQL), custom PL/SQL procedures, and external REST APIs to complete tasks. The framework preserves multi-turn memory, maintaining context across conversations. Together, these capabilities support scalable, context-aware generative AI that integrates with enterprise data and workflows.

The DBMS_CLOUD_AI_AGENT package encapsulates management, orchestration, and security boundaries. See DBMS_CLOUD_AI_AGENT Package for details.

Features of Select AI Agent

The key features include integrated intelligence, flexible tooling, context-aware conversations, and faster deployment.

- **Integrated Intelligence:**
Combines planning, tool use, and reflection so agents can reason about tasks, choose and run tools, observe outcomes, adjust plans, and improve responses throughout conversation. Agents plan steps, run tools, evaluate observations, and update their approach when outcomes miss expectations. This loop strengthens accuracy, reduces rework, and keeps conversations on track.
- **Flexible Tooling:**
Support and use built-in RAG and NL2SQL, custom PL/SQL procedures, and external REST services, without orchestration components or separate infrastructure, so you can keep core logic in the database while integrating external capabilities as needed.
- **Context-Aware Conversations:**
Maintain short-term and long-term memory to keep context across turns, personalize responses, store preferences, and support human-in-the-loop control for corrections and confirmations during multi-turn sessions. Short-term memory keeps the current dialogue coherent. Long-term memory records preferences and prior outcomes, supporting follow-up interactions and oversight by human reviewers.
- **Scalable and Secure:**
Run inside Autonomous AI Database, inherit its security controls, auditing, and performance, reduce data movement, and standardize governance for enterprise deployments and regulated environments at scale. Agents benefit from database security, auditing, and performance characteristics. Keeping processing close to data reduces movement and aligns with governance practices.
- **Faster Development:**

Define agents, tasks, and tools with familiar SQL and PL/SQL, reuse existing procedures, and ship features faster while keeping logic close to operational data and teams without building separate infrastructure.

ReAct Agentic Pattern

Select AI Agent uses ReAct (Reasoning and Acting) agentic pattern where the agent reasons about the request, chooses tools, performs actions, and evaluates results to accomplish a goal.

The following is the pattern for each iteration:

1. **Query:** The user asks a question or states a request. The agent reads it, extracts key details, and prepares to plan the next steps.
2. **Thought and Action:** The agent reasons about options, picks a tool, and runs it to gather data or change state as needed for the task.
3. **Observation:** Observations include tool or query results, confirmation messages, and errors. These become inputs to the agent's next round of reasoning. The agent records observations and checks whether the results support the next step or the final response.
4. **Final Response:** After enough successful thought-action and observations, the agent composes a clear answer, explains important decisions, and shares any next steps or follow-up actions.

Select AI Agent Architecture

Select AI Agent organizes work into four layers: Planning, Tool Use, Reflection, and Memory Management. These layers coordinate reasoning, tool runs, evaluation, and context multi-turn interactions.

Planning: Planning interprets the user request, breaks it into ordered actions, selects candidate tools, and drafts a plan using session context, prior outcomes, and relevant knowledge. The agent analyzes the request, identifies missing details, and proposes an ordered sequence of actions. It chooses tools that fit policy, data scope, and expected outcomes.

Tool Use: Tool Use selects and runs the tool for each action. Supported types include RAG, NL2SQL, custom PL/SQL procedures that can be added when you create a [tool](#), and external REST services such as web search and email. Each step calls a tool with parameters. Built-in tools handle retrieval and SQL generation. Custom PL/SQL encapsulates domain logic. REST tools connect to external services.

Reflection: Reflection evaluates tool results against expectations and proceeds to final response. The agent compares observations to the goal. If results look wrong or if there are tool call errors or user disapproved results, the agent revises reasoning, chooses another tool, or updates the plan before trying again. When results do not fit, it adjusts the plan, selects different tools, or may ask clarifying questions before proceeding. Select AI Agent thoughts can be queried using `USER_CLOUD_AI_CONVERSATION_PROMPTS`. See `DBMS_CLOUD_AI` Views for more information.

Memory Management: Memory Management stores session context and knowledge per agent team. Short-term memory holds recent messages and intermediate results per agent team. Long-term memory records preferences, history, and strategies, improving continuity, personalization, and planning. Long-term memory persists useful knowledge across sessions, improving guidance and response quality over time across agent teams.

Select AI Agent Terminology

Explore the concepts and terms related to Select AI Agent (autonomous agent framework).

The following are the terms related to Select AI Agent:

Term	Definition
Agentic Action	In the context of agents, an action is an instruction that triggers a tool with parameters. The platform runs the tool, handles errors, and returns an observation for the next reasoning step loop. Actions connect planning to execution. They define which tool to run and with which inputs.
Agent	An agent performs tasks for a defined purpose. An agent is a configured worker that reasons about requests, selects tools, runs steps, evaluates results, and produces responses grounded in database context. An agent encapsulates behavior, policies, and available tools for a particular job, such as returns processing or knowledge retrieval.
Agent Team	One or more agents that perform an agentic workflow. A team coordinates responsibilities, shares context, and runs agent-task pairs to complete multi-step interactions reliably. Teams divide work among specialized agents and sequence their contributions. Teams keep shared context and produce a unified response. You can view agent and team run details using the following history views: <code>USER_AI_AGENT_TEAM_HISTORY</code>, <code>USER_AI_AGENT_TASK_HISTORY</code>, and <code>USER_AI_AGENT_TOOL_HISTORY</code>. These views record prompts, responses, and tool runs for each agent, task, and team, providing transparency and traceability across agentic workflows. For more information about these history views, see the <code>DBMS_CLOUD_AI_AGENT</code> History Views.
MCP Server	An MCP Server (Model Context Protocol Server) provides a standardized way for applications to interact with generative AI models and tools by exposing model capabilities, context retrieval, and extensions through a consistent protocol. Using Select AI Agent, you can define tools that can be exposed through the Autonomous AI Database MCP Server. These tools can include built-in tools for SQL generation, RAG, notifications (email and Slack), and websearch, as well as custom tools. Custom tools can expose functions defined in your database.
Observation	The data returned from a tool run: rows, messages, or errors, that the agent records and evaluates during reflection. Observations guide the next reasoning step and provide evidence for the final response.

Term	Definition
Task	A task represents a unit of work. It guides tool selection, parameter mapping, and execution policy, producing results that downstream steps can read and summarize. Tasks specify the goal, inputs, tool choices, and guardrails. They return structured outputs that later steps can consume. You can query the <code>USER_AI_AGENT_TASK_HISTORY</code> view to inspect how tasks are defined, run, and completed within an agentic run. See <code>DBMS_CLOUD_AI_AGENT History Views</code> for more information.
Tool	A tool performs an action such as updating data, retrieving documents, or calling external services. Tools take parameters, can run deterministically or non-deterministically, and return observations for reasoning. Tools encapsulate repeatable operations. They keep side effects controlled and observable, supporting auditing and debugging. In the Select AI Agent framework, tools represent functional building blocks that agents can use during task processing. A tool might be: • A deterministic tool, such as a PL/SQL function, which returns the same output for the same input. • A non-deterministic tool, such as a web search or LLM-based summarization, where outputs may vary depending on timing, network results, or randomness. Agents select tools based on the task's needs and use the output (known as an observation) to reason, decide next steps, or generate responses. Tool metadata and run history are logged to support debugging, observability, and security. You can query the <code>USER_AI_AGENT_TOOL_HISTORY</code> view to monitor tool usage and review run outcomes for each agent task. See the <code>DBMS_CLOUD_AI_AGENT History Views</code> for more information.

Select AI Agent Use Cases

Select AI Agent supports practical scenarios across roles: connect external services, build task-focused automation with PL/SQL tools, and enable conversational access to data using natural-language interactions.

Use cases include:

Integrate external APIs for enriched responses:

Connect REST services such as shipping status, knowledge articles, or messaging. The agent blends external results with database facts and returns timely, actionable guidance.

Build task-specific agents using PL/SQL tools:

Create focused agents that call PL/SQL tools for updates, validations, or transformations, keeping logic close to data, improving performance, and simplifying maintenance across environments consistently. Wrap existing PL/SQL for updates and checks. Agents coordinate tasks, enforce parameters, and record outcomes while staying within transactional controls.

Enable conversational access to enterprise data:

Expose data through natural-language questions grounded on database objects. Use NL2SQL for retrieval, apply policies, and present findings clearly while respecting security and governance requirements. NL2SQL translates questions to SQL on approved objects. Agents summarize results and next steps while following object lists and access rules.

Prerequisites for Using Select AI Agent

Before you use Select AI Agent, you must have privileges to use the `DBMS_CLOUD_AI_AGENT` package and all other privileges that are required for Select AI.

To use Select AI Agent, you must perform the following:

- Perform prerequisites and grant privileges for Select AI. See [Configure AI Profiles Using DBMS_CLOUD_AI](#) for instructions.
-
- At least one agent team defined. See [CREATE_AGENT](#) Procedure.
- Optionally, a `conversation_id` is created to keep track of multi-turn context.

Grant Privileges for Select AI Agent

To configure `DBMS_CLOUD_AI_AGENT`:

Grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI_AGENT` package to the user who wants to use Select AI Agent.

By default, only the system administrator has `EXECUTE` privilege. The administrator can grant `EXECUTE` privilege to other users.

Example of Privileges to Run Select AI Agent

The following example grants the `EXECUTE` privilege to `ADB_USER`:

```
GRANT EXECUTE on DBMS_CLOUD_AI_AGENT to ADB_USER;
```

How Do I Use Select AI Agent

You can use Select AI Agent by using the agent action or the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` function.

You can use Select AI Agent in the following ways:

- Use agent as Select AI action: `Select AI agent <prompt>`.
- Call the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` and supply a prompt within the function along with other parameters.

Both options route the prompt to the chosen agent team and return the team's reply.

See Also:

[#unique_605](#) to learn more.

Examples of Using Select AI Agent

Explore examples that show how to build, configure, and interact with Select AI Agent for common tasks such as movie analysis, log analysis and customer support.

Example: Create an Agent

Create an agent to perform a defined task.

This example creates an agent named `Customer_Return_Agent` responsible for handling product return-related conversations.

This example illustrates using Google as the AI provider as specified in the AI profile named `G00GLE`. The AI profile identifies the LLM the agent uses for reasoning and responses. The `role` attribute provides instructions to guide the agent.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
    agent_name => 'CustomerAgent',
    attributes => '{
      "profile_name": "G00GLE",
      "role": "You are an experienced customer agent who
deals with customers return request."
    }'
  );
END;
/
```

Example: Create Built-In Tools

Create built-in tools such as SQL, RAG, Websearch, Email, and Slack. These tools enable agents and tasks to query data, retrieve knowledge, search the web, and send notifications.

Built-in tools (for example, SQL and RAG) include internal instructions. You can optionally append user-provided instructions to tailor behavior for your use case. Appending an instruction can add task-specific context (for example, scope queries to a specific schema or help enforce a response format). This can improve relevance for specialized scenarios. Poorly written or overly prescriptive instructions can degrade tool performance. Keep instructions short, specific, and consistent with the tool's purpose.

Example: SQL Tool

This example creates a SQL tool that translates natural language queries into SQL statements. SQL tool helps the agents answer data-related questions by mapping prompts into SQL queries.

This example shows using OCI as the AI provider as specified in the AI profile named `nl2sql_profile`. The AI profile identifies the LLM the agent uses for reasoning and responses. In this example, the AI profile `nl2sql_profile` defines the set of SH schema tables that the agent can query, enabling natural language access to commonly used sales history data such as customers, products, promotions, and countries. The SQL tool uses the database objects that are specified in the AI profile, ensuring that generated SQL statements remain accurate and relevant to the SH data set.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'nl2sql_profile',
```

```

        attributes => '{"provider": "oci",
        "credential_name": "GEN1_CRED",
        "oci_compartment_id": "ocid1.compartment.oc1..aaaa..",
        object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}]}

    ');
end;
/

EXEC DBMS_CLOUD_AI_AGENT.DROP_TOOL('SQL');
BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
        tool_name => 'SQL',
        attributes => '{"tool_type": "SQL",
        "tool_params": {"profile_name": "nl2sql_profile"}}'
    );
END;
/

```

Example: RAG Tool

This example creates a RAG (Retrieval Augmented Generation) tool. The RAG tool lets agents retrieve and ground responses in enterprise documents, improving accuracy for knowledge-based answers.

This example illustrates defining a RAG_PROFILE with credentials, a vector index, and specifying profile parameters. Then, creating a vector index RAG_INDEX in Object Storage for document embeddings and creating the RAG_TOOL linked to your profile.

```

BEGIN
    DBMS_CLOUD_AI.CREATE_PROFILE(
        profile_name => 'RAG_PROFILE',
        attributes => '{"provider": "oci",
        "credential_name": "GENAI_CRED",
        "vector_index_name": "RAG_INDEX",
        "oci_compartment_id": "ocid1.compartment.oc1..aaaa..",
        "temperature": 0.2,
        "max_tokens": 3000
        }');
END;
/

BEGIN
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name => 'RAG_INDEX',
        attributes => '{"vector_db_provider": "oracle",
        "location": "https://swiftobjectstorage.us-
        phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
        "object_storage_credential_name": "OCI_CRED",
        "profile_name": "RAG_PROFILE",
        "chunk_overlap": 128,
        "chunk_size": 1024
        }');
END;
/

```

```

    }');
END;
/

EXEC DBMS_CLOUD_AI_AGENT.DROP_TOOL('RAG_TOOL');
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name => 'RAG_TOOL',
    attributes => '{"tool_type": "RAG",
                  "tool_params": {"profile_name": "RAG_PROFILE"}}'
  );
END;
/

```

Example: Websearch Tool

This example creates a Websearch tool for retrieving details from the internet. The Websearch tool enables agents to look up information from the web, such as product prices or descriptions.

This example illustrates adding an ACL entry for the OpenAI provider. Creating credentials OPENAI_CRED with your API key and creating the Websearch tool, describing its purpose, linking it to the credential.

Note

OpenAI is the only Websearch AI provider supported.

```

BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'api.openai.com',
    ace => xs$ace_type(privilege_list => xs$name_list('http'),
                      principal_name => 'ADB_USER',
                      principal_type => xs_acl.ptype_db)
  );
END;
/

BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OPENAI_CRED',
    username        => 'OPENAI',
    password        => '<OPENAI_API_KEY>'
  );
END;
/

EXEC DBMS_CLOUD_AI_AGENT.DROP_TOOL('Websearch');
BEGIN
  DBMS_CLOUD_AI_AGENT.create_tool(
    tool_name => 'Websearch',
    attributes => '{"instruction": "This tool can be used for searching the
details about topics mentioned in notes and prepare a summary about prices,
details on web",
                  "tool_type": "WEBSEARCH",

```

```

                                "tool_params": {"credential_name": "OPENAI_CRED"}}}'
    );
END;
/

```

Example: Notification Tool with Email Type

This example creates an Email notification tool. The Email tool enables agents to send notification emails as part of their workflow.

This example illustrates creating credentials EMAIL_CRED with your password, allowing SMTP access for the database user, and creating a notification tool with type EMAIL, including SMTP details, sender, and recipient addresses.

```

BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'EMAIL_CRED',
    username => '<username>',
    password => '<password>');
END;
/
-- Allow SMTP access for user
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'smtp.email.us-ashburn-1.oci.oraclecloud.com',
    lower_port => 587,
    upper_port => 587,
    ace => xs$ace_type(privilege_list => xs$name_list('SMTP'),
      principal_name => 'ADB_USER',
      principal_type => xs$acl.ptype_db));
END;
/

EXEC DBMS_CLOUD_AI_AGENT.DROP_TOOL('Email');
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name => 'EMAIL',
    attributes => '{"tool_type": "NOTIFICATION",
      "tool_params": {"notification_type" : "EMAIL",
        "credential_name": "EMAIL_CRED",
        "recipient":
          "example_recipient@oracle.com",
          "smtp_host": "smtp.email.us-
            ashburn-1.oci.oraclecloud.com",
          "sender": "example_sender@oracle.com"}}}'
    );
END;
/

```

Example: Notification Tool with Slack Type

This example creates a Slack notification tool. The Slack tool enables agents to deliver notifications directly to a Slack workspace channel.

This example illustrates adding an ACL entry for Slack and creating a notification tool with type SLACK linking it to SLACK_CRED and the target channel.

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
    host      => 'slack.com',
    lower_port => 443,
    upper_port => 443,
    ace       => xs$ace_type(
      privilege_list => xs$name_list('http'),
      principal_name => 'ADB_USER',
      principal_type => xs_acl.ptype_db));
END;
/

BEGIN
  DBMS_CLOUD_AI_AGENT.create_tool(
    tool_name => 'slack',
    attributes => '{"tool_type": "SLACK",
                  "tool_params": {"credential_name": "SLACK_CRED",
                                  "channel": "<channel_number>"} }'
  );
END;
/
```

Example: Create a Task

Create a task an agent can perform.

Note

Only a DBA can grant EXECUTE privileges and network ACL procedure.

Example: Create an Email Task

This example creates Generate_Email_Task that instructs the LLM to produce a standard confirmation email using structured data.

The following example illustrates using the instruction attribute and providing instructions on what the email should include.

```
BEGIN DBMS_CLOUD_AI_AGENT.DROP_TASK('Generate_Email_Task');
EXCEPTION WHEN OTHERS THEN NULL; END;
/

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'Generate_Email_Task',
    attributes => '{"instruction": "Use the customer information and product
details to generate an email in a professional format. The email should: ' ||
'1. Include a greeting to the customer by name' ||
'2. Specify the item being returned, the order number,
and the reason for the return' ||
'3. If it is a refund, state the refund will be issued to
the credit card on record.' ||
```



```

        '4. If it is a replacement, state that the replacement
will be shipped in 3-5 business days.}"'
    );
END;

```

Example: Create a Fetch Log Task

This example creates a `FETCH_LOGS_TASK` that retrieves logs based on the request.

This example illustrates creating a task that uses the `log_fetcher` tool to retrieve logs. This custom tool specifies the PL/SQL procedure `fetch_logs`. The task instruction specifies what the task needs accomplish. The attribute `enable_human_tool` is set to `true` so a person can step in to guide or approve the task flow if needed. See [#unique_607](#) for a complete example.

```

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name =>'FETCH_LOGS_TASK',
    attributes =>'{
      "instruction": "Fetch the log entries from the database based on user
request: {query}",
      "tools": ["log_fetcher"],
      "enable_human_tool" : "true"
    }'
  );
END;
/

```

Example: Create a Log Analysis Task

This example creates a `ANALYZE_LOG_TASK` that analyzes the fetched logs.

The following example illustrates using `FETCH_LOGS_TASK` as the input attribute. The task named `ANALYZE_LOG_TASK` starts after logs have been gathered to have `FETCH_LOGS_TASK` output as an input. The task instruction specifies what the task needs accomplish. The attribute `enable_human_tool` is set to `false` indicating the absence of human-in-the-loop review.

```

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name =>'ANALYZE_LOG_TASK',
    attributes =>'{
      "instruction": "Analyze the fetched log entries
retrieved based on user request: {query} ' ||
      'Generate a detailed report include issue analysis and
possible solution.",
      "input" : "FETCH_LOGS_TASK",
      "enable_human_tool" : "false"
    }'
  );
END;
/

```

Example: Create an Agent Team

Create agent teams to accomplish your tasks.

Example: Create an Agent Team

This example creates the `ReturnAgency` team and includes a single agent `Customer_Return_Agent`. The task `Return_And_Price_Match` is assigned to the agent. This

task manages return requests by asking for the reason and updating the order status in your database.. The process is set to `sequential` to run the tasks in a defined order.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TEAM(
    team_name =>
'ReturnAgency',
    attributes => '{"agents": [{"name":"Customer_Return_Agent","task" :
"Return_And_Price_Match"}]},
                  "process":
"sequential"}');

END;
/
```

See [#unique_608](#) for a complete example.

Example: Create Multi-Agent Team

This example creates the `Ops_Issues_Solution_Team` team and includes two agents to handle log analysis and troubleshooting: `Data_Engineer` and `Ops_Manager`. The `Data_Engineer` agent runs the task `fetch_logs_task` and `Ops_Manager` agent runs the task `analyze_log_task`. The process is set to `sequential` to run the tasks in a defined order.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.create_team(
    team_name => 'Ops_Issues_Solution_Team',
    attributes => '{"agents": [{"name":"Data_Engineer","task" :
"fetch_logs_task"},
                        {"name":"Ops_Manager","task" :
"analyze_log_task"}]},
                  "process": "sequential"
                  }');

END;
/
```

See [#unique_607](#) for a complete example.

Example: Create a Movie Analysis Agent with Built-In Tools

This example shows how you can create a movie analysis agent using Select AI Agent. In this example, you set up a movie analysis agent that retrieves data, answers questions, searches the web, and emails the analysis or sends Slack notifications.

The following example assumes that the data is available to you.

This example creates a `MOVIE_ANALYST` agent and uses multiple built-in tools such as `SQL`, `RAG`, `WEBSEARCH`, and `NOTIFICATION` tool of type: `EMAIL` and `SLACK`. The agent answers movie related questions using natural language prompts.

In this example, after your DBA grants `EXECUTE` privileges for: the `DBMS_CLOUD_AI_AGENT`, `DBMS_CLOUD_AI`, `DBMS_CLOUD_PIPELINE` packages, `ACL` access for your AI provider, `SMTP` access, and `Slack` access, you begin by creating the `MOVIE_ANALYST` agent.

Create an Agent

You create an agent called `MOVIE_ANALYST` with a profile (`GROK`) and role of answering questions about movies, actors, and genres.

Create Task

You create a task called `ANALYZE_MOVIE_TASK` with instructions to answer movie-related queries.

Create Team

You create a `MOVIE_AGENT_TEAM` team with `MOVIE_ANALYST` as the agent and the task as `ANALYZE_MOVIE_TASK` and set it as an active team.

You can later attach tools such as SQL, RAG, Websearch, or Notification to extend its capabilities.

Run the Select AI Agent Team

You now run the agent team by using `select ai agent` as a prefix to your prompts.

Create Tools

You then attach different built-in agent tools:

- **SQL:** Uses an NL2SQL profile to translate questions into SQL queries and other supported Select AI actions.
- **RAG:** Retrieve knowledge-based context from stored documents.
- **WEBSEARCH:** Gather movie details or prices online.
- **NOTIFICATION:**
 - **EMAIL:** Send answers to user prompts, movie reports, or reviews to a recipient.
 - **SLACK:** Send answers to user prompts, summaries, or updates directly to Slack.

The complete example is as follows:

```
--Grants EXECUTE privilege to ADB_USER
--
GRANT EXECUTE on DBMS_CLOUD_AI_AGENT to ADB_USER;
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;
GRANT EXECUTE on DBMS_CLOUD_PIPELINE to ADB_USER;

-- Websearch tool accesses OPENAI endpoint, allow ACL access
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'api.openai.com',
    ace => xs$ace_type(privilege_list => xs$name_list('http'),
                      principal_name => 'ADB_USER',
                      principal_type => xs$acl.ptype_db)
  );
END;
/

-- To allow Email tool in Autonomous Database, allow SMTP access
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'smtp.email.us-ashburn-1.oci.oraclecloud.com',
    lower_port => 587,
    upper_port => 587,
    ace => xs$ace_type(privilege_list => xs$name_list('SMTP'),
                      principal_name => 'ADB_USER',
```

```

                                principal_type => xs_acl.ptype_db));
END;
/

-- Allow ACL access to Slack

BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE (
    host          => 'slack.com',
    lower_port    => 443,
    upper_port    => 443,
    ace           => xs$ace_type(
      privilege_list => xs$name_list('http'),
      principal_name => 'ADB_USER',
      principal_type => xs_acl.ptype_db));
END;
/

PL/SQL procedure successfully completed.

--Create an agent
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
    agent_name => 'MOVIE_ANALYST',
    attributes => '{"profile_name": "GROK",
                  "role": "You are an AI Movie Analyst. Your can help
answer a variety of questions related to movies. "
                  }'
  );
END;
/

PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'ANALYZE_MOVIE_TASK',
    attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}",
                  "enable_human_tool" : "true"
                  }'
  );
END;
/

PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TEAM(
    team_name =>
'MOVIE_AGENT_TEAM',

    attributes => '{"agents": [{"name": "MOVIE_ANALYST", "task" :
"ANALYZE_MOVIE_TASK"}],
                  "process": "sequential"
                  }');

```

```
END;
/
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');
```

PL/SQL procedure successfully completed.

```
select ai agent who are you?;
```

RESPONSE

```
-----
--
I'm MOVIE_ANALYST, an AI Movie Analyst here to assist with any questions or
topi
cs related to movies. Whether you need information on films, actors,
directors,
genres, or recommendations, I'm ready to help. What can I assist you with
regard
ing movies?

-- SQL TOOL
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name
=>'nl2sql_profile',

    attributes => '{"provider":
"oci",
  "credential_name": "GENAI_CRED",
  "oci_compartment_id" :
"ocid1.compartment.oc1..aaaaa...",
  "object_list": [{"owner": "ADB_USER", "name": "GENRE"},
                  {"owner": "ADB_USER", "name": "CUSTOMER"},
                  {"owner": "ADB_USER", "name": "WATCH_HISTORY"},
                  {"owner": "ADB_USER", "name": "STREAMS"},
                  {"owner": "ADB_USER", "name": "MOVIES"},
                  {"owner": "ADB_USER", "name": "ACTORS"}]
  }');
END;
/
```

PL/SQL procedure successfully completed.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name => 'SQL',
    attributes => '{"tool_type": "SQL",
                  "tool_params": {"profile_name": "nl2sql_profile"}}'
  );
END;
/
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI_AGENT.drop_task('ANALYZE_MOVIE_TASK');
```

PL/SQL procedure successfully completed.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'ANALYZE_MOVIE_TASK',
    attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                                'You can use SQL tool to search the data from
database",
    "tools": ["SQL"],
    "enable_human_tool" : "true"
  }'
  );
END;
/
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');
```

PL/SQL procedure successfully completed.

```
-- SQL tool retrieves the movie with the highest popularity view count from
the watch_history table
select ai agent what is the most popular movie?;
```

RESPONSE

```
-----
The most popular movie is "Laugh Out Loud" released in 2008.
```

```
-- RAG TOOL
```

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'RAG_PROFILE',
    attributes => '{"provider": "oci",
    "credential_name": "GENAI_CRED",
    "vector_index_name": "RAG_INDEX",
    "oci_compartment_id": "ocid1.compartment.oc1..aaaaa...",
    "temperature": 0.2,
    "max_tokens": 3000
  }'
  );
END;
/
```

PL/SQL procedure successfully completed.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
    index_name => 'RAG_INDEX',
```

```

        attributes => '{"vector_db_provider": "oracle",
                        "location": "https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
                        "object_storage_credential_name": "MY_OCI_CRED",
                        "profile_name": "RAG_PROFILE",
                        "vector_dimension": 1024,
                        "vector_distance_metric": "cosine",
                        "chunk_overlap":128,
                        "chunk_size":1024
                    }');
END;
/
PL/SQL procedure successfully completed.

```

```

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name => 'RAG_TOOL',
    attributes => '{"tool_type": "RAG",
                  "tool_params": {"profile_name": "RAG_PROFILE"}}'
  );
END;
/

```

PL/SQL procedure successfully completed.

```

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'ANALYZE_MOVIE_TASK',
    attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                  'You can use RAG tool to search the information from
the knowledge base user give.",
                  "tools": ["RAG_TOOL"],
                  "enable_human_tool" : "true"
                  }'
  );
END;
/

```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');
```

PL/SQL procedure successfully completed.

```
-- Rag seach the object store to find review or comments of Movie Laugh out
Loud
select ai agent Please find the comments of Movie Laugh out Loud;
```

RESPONSE

```
-----
-----
```

```

-----
-----
-----
-----
-----

```

The comments for the movie "Laugh Out Loud" (2008) are as follows:

1. A lighthearted comedy that delivers plenty of laughs, though the plot is fairly predictable.
2. The performances are fun, especially from the lead actor who keeps the energy high.
3. Some jokes feel a bit outdated, but overall it is an enjoyable watch for a casual movie night.
4. Good chemistry between the cast members, which makes the humor more natural.
5. Not a groundbreaking comedy, but it does what it promises makes you laugh out loud.

```

-- WEBSEARCH TOOL
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'OPENAI_CRED',
    username        => 'OPENAI',
    password        => '<API_KEY>'
  );
END;
/

```

PL/SQL procedure successfully completed.

```

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
    tool_name => 'WEBSEARCH_TOOL',
    attributes => '{"instruction": "This tool can be used for searching the
details about topics mentioned in notes and prepare a summary about prices,
details on web",
                  "tool_type": "WEBSEARCH",
                  "tool_params": {"credential_name": "OPENAI_CRED"}}'
  );
END;
/

```

PL/SQL procedure successfully completed.

```

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'ANALYZE_MOVIE_TASK',
    attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                  'You can use WEBSEARCH_TOOL tool to search the
information from internet.",
                  "tools": ["WEBSEARCH_TOOL"],
                  "enable_human_tool" : "true"
                  }'
  );
END;
/

```



```

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'ANALYZE_MOVIE_TASK',
    attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                                'You can use EMAIL TOOL tool to send email to the
user."',
    "tools": ["EMAIL"],
    "enable_human_tool" : "true"
  );
END;
/

```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;
```

PL/SQL procedure successfully completed.

```
EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('MOVIE_AGENT_TEAM');
```

PL/SQL procedure successfully completed.

```
select ai agent Please help me write a review of Movie "Barbie" and send the
review to my email;
```

RESPONSE

```

-----
-----
-----

```

I have written a review for the movie "Barbie" (2023) and sent it to your email. Please check your inbox for the detailed review. If you have any additional feedback or would like me to revise the review, let me know!

-- NOTIFICATION TOOL WITH SLACK TYPE

```

DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
  tool_name => 'SLACK_TOOL',
  attributes => '{"tool_type": "SLACK",
                                "tool_params": {"credential_name": "SLACK_CRED",
                                                "channel": "<channel_number>"}}'
);
END;
/

```

PL/SQL procedure successfully completed.

```

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'ANALYZE_MOVIE_TASK',
    attributes => '{"instruction": "Help the user with their request about
movies. User question: {query}. ' ||
                                'You can use SLACK TOOL to send the notification to
the user."',
    "tools": ["SLACK_TOOL"],

```

```

        "enable_human_tool" : "true"
    },
);
END;
/

```

PL/SQL procedure successfully completed.

select ai agent Please help me find the top 3 most-watched movies of 2023 and send them to me on slack;

RESPONSE

```

-----
-----
I have sent the list of the top 3 most-watched movies of 2023 to you via
Slack. Please check your Slack notifications for the details.

```

Example: Create a Product Return Agent

This example shows how you can create a multi-turn conversational agent using Select AI Agent. In this example, you set up a customer-service agent that handles product returns and updates the return status in your database.

This example creates an agent named *Customer_Return_Agent* and a tool named *Update_Order_Status_Tool* and then defines a task and a team to handle product return.

In this example, after your DBA grants EXECUTE privileges for: the DBMS_CLOUD_AI_AGENT and DBMS_CLOUD_AI, you begin by creating a sample data of customers, customer order status, and a function to update the customer order status. Then, create an agent named *Customer_Return_Agent*.

Create an Agent

You create an agent called *Customer_Return_Agent* with a profile (OCI_GENAI_GROK) and role to manage return requests.

Create Tools

You then create an agent tool named *Update_Order_Status_Tool* to update the status of the order in your database.

Create Task

You create a task called *Handle_Product_Return_Task* to guide the flow: ask for the reason (no longer needed, arrived too late, box broken, or defective). Proceed with a defective return flow.

Create Team

You create an agent team called *Return_Agency_Team* with *Customer_Return_Agent* as the agent and set it as an active team.

Run the Select AI Agent Team

You now run the agent team by using `select ai agent` as a prefix to your prompts.

The complete example is as follows:

```
--Grants EXECUTE privilege to ADB_USER
--
GRANT EXECUTE on DBMS_CLOUD_AI_AGENT to ADB_USER;
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
    agent_name => 'Customer_Return_Agent',
    attributes => '{"profile_name": "OCI_GENAI_GROK",
                  "role": "You are an experienced customer return agent who
deals with customers return requests."}');
END;
/
```

PL/SQL procedure successfully completed.

```
--Sample customer data
CREATE TABLE CUSTOMERS (
  customer_id  NUMBER(10) PRIMARY KEY,
  name         VARCHAR2(100),
  email        VARCHAR2(100),
  phone        VARCHAR2(20),
  state        VARCHAR2(2),
  zip          VARCHAR2(10)
);

INSERT INTO CUSTOMERS (customer_id, name, email, phone, state, zip) VALUES
(1, 'Alice Thompson', 'alice.thompson@example.com', '555-1234', 'NY',
'10001'),
(2, 'Bob Martinez', 'bob.martinez@example.com', '555-2345', 'CA', '94105'),
(3, 'Carol Chen', 'carol.chen@example.com', '555-3456', 'TX', '73301'),
(4, 'David Johnson', 'david.johnson@example.com', '555-4567', 'IL', '60601'),
(5, 'Eva Green', 'eva.green@example.com', '555-5678', 'FL', '33101');

--create customer order status
CREATE TABLE CUSTOMER_ORDER_STATUS (
  customer_id  NUMBER(10),
  order_number  VARCHAR2(20),
  status        VARCHAR2(30),
  product_name  VARCHAR2(100)
);

INSERT INTO CUSTOMER_ORDER_STATUS (customer_id, order_number, status,
product_name) VALUES
(2, '7734', 'delivered', 'smartphone charging cord'),
(1, '4381', 'pending_delivery', 'smartphone protective case'),
(2, '7820', 'delivered', 'smartphone charging cord'),
(3, '1293', 'pending_return', 'smartphone stand (metal)'),
(4, '9842', 'returned', 'smartphone backup storage'),
(5, '5019', 'delivered', 'smartphone protective case'),
(2, '6674', 'pending_delivery', 'smartphone charging cord'),
```

```

(1, '3087', 'returned', 'smartphone stand (metal)'),
(3, '7635', 'pending_return', 'smartphone backup storage'),
(4, '3928', 'delivered', 'smartphone protective case'),
(5, '8421', 'pending_delivery', 'smartphone charging cord'),
(1, '2204', 'returned', 'smartphone stand (metal)'),
(2, '7031', 'pending_delivery', 'smartphone backup storage'),
(3, '1649', 'delivered', 'smartphone protective case'),
(4, '9732', 'pending_return', 'smartphone charging cord'),
(5, '4550', 'delivered', 'smartphone stand (metal)'),
(1, '6468', 'pending_delivery', 'smartphone backup storage'),
(2, '3910', 'returned', 'smartphone protective case'),
(3, '2187', 'delivered', 'smartphone charging cord'),
(4, '8023', 'pending_return', 'smartphone stand (metal)'),
(5, '5176', 'delivered', 'smartphone backup storage');

--Create a update customer order status function
CREATE OR REPLACE FUNCTION UPDATE_CUSTOMER_ORDER_STATUS (
    p_customer_name IN VARCHAR2,
    p_order_number  IN VARCHAR2,
    p_status        IN VARCHAR2
) RETURN CLOB IS
    v_customer_id  customers.customer_id%TYPE;
    v_row_count    NUMBER;
BEGIN
    -- Find customer_id from customer_name
    SELECT customer_id
    INTO v_customer_id
    FROM customers
    WHERE name = p_customer_name;

    UPDATE customer_order_status
    SET status = p_status
    WHERE customer_id = v_customer_id
      AND order_number = p_order_number;

    v_row_count := SQL%ROWCOUNT;

    IF v_row_count = 0 THEN
        RETURN 'No matching record found to update.';
    ELSE
        RETURN 'Update successful.';
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RETURN 'Error: ' || SQLERRM;
END;

--Create Tool
BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
        tool_name => 'Update_Order_Status_Tool',
        attributes => '{"instruction": "This tool updates the database to
reflect return status change. Always confirm user name and order number with
user before update status",
                        "function" : "update_customer_order_status"}',
        description => 'Tool for updating customer order status in database

```

```

table.'
);
END;
/
--Create Task
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'Handle_Product_Return_Task',
    attributes => '{"instruction": "Process a product return request from a
customer:{query}'} ||
    '1. Ask customer the order reason for return (no longer
needed, arrived too late, box broken, or defective)' ||
    '2. If no longer needed:' ||
    '  a. Inform customer to ship the product at their
expense back to us.' ||
    '  b. Update the order status to return_shipment_pending
using Update_Order_Status_Tool.' ||
    '3. If it arrived too late:' ||
    '  a. Ask customer if they want a refund.' ||
    '  b. If the customer wants a refund, then confirm
refund processed and update the order status to refund_completed' ||
    '4. If the product was defective or the box broken:' ||
    '  a. Ask customer if they want a replacement or a
refund' ||
    '  b. If a replacement, inform customer replacement is
on its way and they will receive a return shipping label for the defective
product, then update the order status to replaced' ||
    '  c. If a refund, inform customer to print out the
return shipping label for the defective product, return the product, and
update the order status to refund' ||
    '5. After the completion of a return or refund, ask if
you can help with anything else.' ||
    '  End the task if user does not need help on anything
else",
    "tools": ["Update_Order_Status_Tool"]}]'
  );
END;
/
SQL> BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TEAM(
    team_name =>
'Return_Agency_Team',

    attributes => '{"agents": [{"name":"Customer_Return_Agent","task" :
"Handle_Product_Return_Task"}],
"process":
"sequential"}');

END;
/

PL/SQL procedure successfully completed.

SQL> EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('Return_Agency_Team');

PL/SQL procedure successfully completed.

```

```
--Validate that the agent team is set in the current session with get_team
SQL> SELECT DBMS_CLOUD_AI_AGENT.GET_TEAM from DUAL;
```

```
--Interact with the agent
```

```
SQL> select ai agent I want to return a smartphone case;
```

RESPONSE

Could you please tell me the reason for returning the smartphone case? Is it no longer needed, arrived too late, box broken, or defective?

```
SQL> select ai agent the item is defective
```

RESPONSE

I'm sorry to hear that the smartphone case is defective. Would you like a replacement or a refund?

```
SQL> select ai agent I will need a replacement
```

RESPONSE

Can you please confirm your name and order number so I can update the status and process the replacement?

```
SQL> select ai agent I'm Bob Martinez and my order number is 7820
```

RESPONSE

Is there anything else I can help you with?

```
SQL> select ai agent No, I'm all set. Thanks
```

RESPONSE

Thank you for reaching out, Bob. I'm glad I could assist with the return of your smartphone case and stand. If you have any further questions or need assistance in the future, feel free to contact us. Have a great day!

Example: Use Select AI Agent with Database Actions and APEX

If you are using web SQL clients such as **Database Actions**, run the agent team by using the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` function and supply your prompt within the function.

Note

Do not use `DBMS_CLOUD_AI_AGENT.SET_TEAM` in Database Actions or APEX Service. Instead, specify the agent team using the `team_name` argument of `DBMS_CLOUD_AI_AGENT.RUN_TEAM`.

The following example creates a conversation ID to track the customer conversation history and then uses the `DBMS_CLOUD_AI_AGENT.RUN_TEAM` function to interact with the product return agent.

See [#unique_609](#) for more information.

```
CREATE OR REPLACE PACKAGE my_globals IS
  l_team_cov_id varchar2(4000);
```

```

END my_globals;
/
-- Create conversation
DECLARE
    l_team_cov_id varchar2(4000);
BEGIN
    l_team_cov_id := DBMS_CLOUD_AI.create_conversation();
    my_globals.l_team_cov_id := l_team_cov_id;
    DBMS_OUTPUT.PUT_LINE('Created conversation with ID: ' ||
my_globals.l_team_cov_id);
END;

--Interact with the agent
DECLARE
    v_response VARCHAR2(4000);
BEGIN
    v_response := DBMS_CLOUD_AI_AGENT.RUN_TEAM(
        team_name => 'Return_Agency_Team',
        user_prompt => 'I want to return a smartphone case',
        params      => '{"conversation_id": "' || my_globals.l_team_cov_id || '"'
    );
    DBMS_OUTPUT.PUT_LINE(v_response);
END;

```

Example: Fetch and Analyze Log Reports

This example shows how you can create an agent team using two agents-task pairs to retrieve logs and analyze them.

This example creates:

- a two-agent workflows for troubleshooting: *Data_Engineer* and *Ops_Manager*
- one tool: log-fetcher

and then defines tasks and team to fetch logs and analyze them.

Note

Only a DBA can run EXECUTE privileges and network ACL procedure.

In this example, after your DBA grants EXECUTE privileges for: the DBMS_CLOUD_AI_AGENT package, DBMS_CLOUD_AI package, and ACL access for your AI providers, you begin by creating the Data_Engineer and Ops_Manager agents.

Create an Agent

You create the agent called Data_Engineer with a profile (GOOGLE) that uses Google as the AI provider and role to retrieve and process complex data.

Create the agent called Ops_Manager with a profile (OPENAI) that uses OpenAI as the AI provider and role to analyze the data.

Create Tool

You then create an agent tool: log_fetcher: returns log entries after a given date. This uses a custom procedure fetch_logs.

Create Tasks

You define two tasks `fetch_logs_task` and `analyze_log_task` to guide the flow. The `fetch_logs_task` calls `log_fetcher` to retrieve logs based on the request. The `analyze_log_task` analyzes the fetched logs.

Create Team

You create `Ops_Issues_Solution_Team` team with `Data_Engineer` and `Ops_Manager` to run sequentially.

Run the Select AI Agent

You now set the active team and run the agent team with `select ai agent` as a prefix to your prompts.

The complete example is as follows:

```
--Grants EXECUTE privilege to ADB_USER
--
GRANT EXECUTE on DBMS_CLOUD_AI_AGENT to ADB_USER;
GRANT EXECUTE on DBMS_CLOUD_AI to ADB_USER;

-- Grant Network ACL for OpenAI endpoint
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'api.openai.com',
        ace => xs$ace_type(privilege_list => xs$name_list('http'),
                           principal_name => 'ADB_USER',
                           principal_type => xs_acl.ptype_db)
    );
END;
/
--Grant Network ACL for Google endpoint
BEGIN
    DBMS_NETWORK_ACL_ADB_USER.APPEND_HOST_ACE(
        host => 'generativelanguage.googleapis.com',
        ace => xs$ace_type(privilege_list => xs$name_list('http'),
                           principal_name => 'ADB_USER',
                           principal_type => xs_acl.ptype_db)
    );
END;
/
PL/SQL procedure successfully completed.

--Create a table with Database logs and insert sample data

SQL> CREATE TABLE app_logs (
    log_id          NUMBER GENERATED BY DEFAULT AS IDENTITY,
    log_timestamp   DATE,
    log_message     VARCHAR2(4000)
);

Table created.

SQL> INSERT INTO app_logs (log_timestamp, log_message) VALUES (
    TO_DATE('2025-03-22 03:15:45', 'YYYY-MM-DD HH24:MI:SS'),
    'INFO: Database Cluster: Failover completed successfully. Standby promoted
```

```
to primary. Downtime duration: 33 seconds. Correlation ID: dbfailover102.'
);
```

```
1 row created.
```

```
SQL> INSERT INTO app_logs (log_timestamp, log_message) VALUES (
    TO_DATE('2025-03-23 08:44:10', 'YYYY-MM-DD HH24:MI:SS'),
    'INFO: Switchover Process: Synchronization restored. Performing scheduled
    switchover. Correlation ID: dbswitchover215.'
);
```

```
1 row created.
```

```
SQL> INSERT INTO app_logs (log_timestamp, log_message) VALUES (
    TO_DATE('2025-03-24 03:15:12', 'YYYY-MM-DD HH24:MI:SS'),
    'ERROR: Database Cluster: Primary database unreachable, initiating failover
    to standby. Correlation ID: dbfailover102.'
);
```

```
1 row created.
```

```
SQL> COMMIT;
```

```
Commit complete.
```

```
-- create data engineer agent
```

```
SQL> BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
        agent_name => 'Data_Engineer',
        attributes => '{"profile_name": "GOOGLE",
                      "role": "You are a specialized data ingestion
engineer with expertise in ' ||
                      'retrieving and processing data from complex
database systems."
                      }'
    );
END;
/
```

```
PL/SQL procedure successfully completed.
```

```
-- create ops manager agent
```

```
SQL> BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
        agent_name => 'Ops_Manager',
        attributes => '{"profile_name": "OPENAI",
                      "role": "You are an experienced Ops manager who
excels at analyzing ' ||
                      'complex log data, diagnosing the issues."
                      }'
    );
END;
/
```

```
PL/SQL procedure successfully completed.
```

```
-- create log_fetcher tool
-- fetch_logs is a customized procedure.
-- Please make sure you have created your own procedure before use it in the
tool

--Create a customized fetch_logs procedure
SQL> CREATE OR REPLACE FUNCTION fetch_logs(since_date IN DATE) RETURN CLOB IS
    l_logs CLOB;
BEGIN
    SELECT JSON_ARRAYAGG(log_message RETURNING CLOB)
        INTO l_logs
        FROM app_logs
        WHERE log_timestamp >= since_date
        ORDER BY log_timestamp;

    RETURN l_logs;
EXCEPTION
    WHEN OTHERS THEN
        RETURN 'Error fetching logs: ' || SQLERRM;
END fetch_logs;
/
```

Function created.

```
SQL> BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TOOL(
        tool_name => 'log_fetcher',
        attributes => '{"instruction": "retrieves and returns all log
messages with a LOG_TIMESTAMP greater than or equal to the input date.",
                        "function": "fetch_logs"}'
    );
END;
/
```

PL/SQL procedure successfully completed.

```
-- create task with log fetcher tool
```

```
SQL> BEGIN
    DBMS_CLOUD_AI_AGENT.CREATE_TASK(
        task_name => 'fetch_logs_task',
        attributes => '{"instruction": "Fetch the log entries from the
database based on user request: {query}}.",
                        "tools": ["log_fetcher"]}'
    );
END;
/
```

PL/SQL procedure successfully completed.

```
-- create task with predefined rag and slack tool
```

```
SQL> BEGIN
    DBMS_CLOUD_AI_AGENT.create_task(
        task_name => 'analyze_log_task',
        attributes => '{"instruction": "Analyze the fetched log entries
```

```

retrieved based on user request: {query} ' ||
        'Generate a detailed report include issue analysis and
possible solution.",
        "input" : "fetch_logs_task"
    }'
);
END;
/

```

PL/SQL procedure successfully completed.

```

-- create team and set agent team
SQL> BEGIN
    DBMS_CLOUD_AI_AGENT.create_team(
        team_name => 'Ops_Issues_Solution_Team',
        attributes => '{"agents": [{"name":"Data_Engineer","task" :
"fetch_logs_task"},
                                {"name":"Ops_Manager","task" :
"analyze_log_task"}]},
        "process": "sequential"
    }');
END;
/

```

PL/SQL procedure successfully completed.

```
SQL> EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('Ops_Issues_Solution_Team');
```

PL/SQL procedure successfully completed.

```
SQL> select ai agent fetch and analyze the logs after 03/15 2025;
```

RESPONSE

```

-----
1. Issue: High volume of 500 Internal Server Errors between 03/22 and 03/24.
   Solution: Review server application logs to identify failing components;
add better exception handling and fallback mechanisms to prevent service
crashes.

2. Issue: Increased response time on /api/v1/user and /checkout, peaking on
03/25.
   Solution: Profile backend queries and services, apply caching where
applicable, and offload static content to a CDN.

3. Issue: Detected brute-force login attack with over 500 failed POST
attempts from a single IP.
   Solution: Add rate-limiting, temporary IP bans, and CAPTCHA on the /login
endpoint to prevent credential stuffing.

4. Issue: Suspicious User-Agent headers such as curl/7.58.0 or empty headers
mimicking mobile devices.
   Solution: Block malformed or uncommon User-Agent strings and log them for
threat intelligence analysis.

```

5. Issue: DDoS-like traffic spike from distributed IPs observed on 03/20.

Solution: Enable DDoS protection at the CDN or cloud provider level, and configure autoscaling to absorb burst traffic.

Example: Create a Customized HTTP Tool

This example defines a PL/SQL function that sends an HTTP GET request and returns the response text. The function is then added in the Select AI Agent tool.

You can create a custom tool that sends a REST request using the `DBMS_CLOUD.SEND_REQUEST` procedure and returns the response as text. This example shows how to write the PL/SQL function and add it as a tool for use in agent tasks.

The function `get_url_content` sends a GET request to the target URL and returns the response as a CLOB. The `DBMS_CLOUD_AI_AGENT.CREATE_TOOL` uses the `get_url_content` function and creates a tool named `HTTP_TOOL` that Select AI agent can call when a task requires fetching content from a URL.

```
CREATE OR REPLACE FUNCTION get_url_content (
    url IN CLOB
) RETURN CLOB
AS
    l_resp    DBMS_CLOUD_TYPES.RESP;
    l_result  CLOB;
BEGIN
    l_resp := DBMS_CLOUD.SEND_REQUEST(
        credential_name => NULL,
        method          => 'GET',
        uri              => url
    );
    l_result :=
DBMS_VECTOR_CHAIN.UTL_TO_TEXT(DBMS_CLOUD.GET_RESPONSE_RAW(l_resp);
    RETURN l_result;
END get_url_content;
/

BEGIN
    DBMS_CLOUD_AI_AGENT.create_tool(
        tool_name => 'HTTP_TOOL',
        attributes => '{
            "instruction": "This tool fetches and returns the plain text content
from the specified URL. ",
            "function": "get_url_content"
        }'
    );
END;
/
```

Example: View Agent Prompts and Responses from the Latest Team Run

This example helps debug recent Select AI Agent activity by displaying prompts, responses, and agent thoughts from the most recent agent team run using historical views and conversation data.

This example retrieves and displays detailed logs from the most recent Select AI Agent team run. The example uses `USER_AI_AGENT_TEAM_HISTORY`, `USER_AI_AGENT_TASK_HISTORY`, and `USER_CLOUD_AI_CONVERSATION_PROMPTS` views to show each agent's team name, task, associated prompts, and responses. Use this query to troubleshoot issues by reviewing both the input prompt and the generated responses from the latest team run.

```
WITH latest_team AS (
  SELECT team_exec_id, team_name, start_date
  FROM user_ai_agent_team_history
  ORDER BY start_date DESC
  FETCH FIRST 1 ROW ONLY
),
latest_task AS (
  SELECT team_exec_id, task_name, agent_name, conversation_params,
  start_date,
  ROW_NUMBER() OVER (PARTITION BY team_exec_id, task_name,
  agent_name
  ORDER BY start_date DESC) as rn
  FROM user_ai_agent_task_history
)
SELECT
  team.team_name,
  task.task_name,
  task.agent_name,
  p.prompt,
  p.prompt_response
FROM latest_team team
JOIN latest_task task
  ON team.team_exec_id = task.team_exec_id
  AND task.rn = 1
LEFT JOIN user_cloud_ai_conversation_prompts p
  ON p.conversation_id = JSON_VALUE(task.conversation_params,
  '$.conversation_id')
ORDER BY task.start_date DESC NULLS LAST,
  p.created DESC NULLS LAST;
```

```
USER_AI_AGENT_TOOL_HISTORYUSER_AI_AGENT_TASK_HISTORYUSER_AI_AGENT_TASK_HISTORY
```

```
--View the tool history
```

```
select * from USER_AI_AGENT_TOOL_HISTORY
order by START_DATE desc
```

```
--View the task history
```

```
select * from USER_AI_AGENT_TASK_HISTORY
order by START_DATE desc
```

```
--View the team history
```

```
select * from USER_AI_AGENT_TEAM_HISTORY
order by START_DATE desc
```

Example: Resume an Agent Team Run from WAITING_FOR_HUMAN State

This example shows how you can resume an agent team run when the task history status is WAITING_FOR_HUMAN. When an agent team enters the WAITING_FOR_HUMAN state, it pauses until a user provides the next input or confirmation. You can resume the run depending on how you use the agent team: on SQL command line or through DBMS_CLOUD_AI_AGENT.RUN_TEAM.

Example: Resume an Agent Team Run Using SQL Command Line

If you are using SELECT AI_AGENT <prompt> on the SQL command line, the Select AI Agent automatically resumes the paused team run. The conversation context, including prior prompts and reasoning, is maintained without requiring additional parameters.

Example: Resume an Agent Team Run Using DBMS_CLOUD_AI_AGENT.RUN_TEAM

If you are using the DBMS_CLOUD_AI_AGENT.RUN_TEAM procedure, you can resume a paused agent team by passing the same conversation_id that was used in the initial run. The agent team run state, including pending actions, and context, is preserved and the agent team continues exactly from where it stopped.

See [#unique_608](#) for an example using the DBMS_CLOUD_AI_AGENT.RUN_TEAM function.

```
--Interact with the agent
DECLARE
  v_response VARCHAR2(4000);
BEGIN
  v_response := DBMS_CLOUD_AI_AGENT.RUN_TEAM(
    team_name    => '<same initial team name>',
    user_prompt  => 'response to request',
    params       => '{"conversation_id": "<same initial conversation_id>"}'
  );
  DBMS_OUTPUT.PUT_LINE(v_response);
END;
```

5

Use Cases

Configuration Use Cases

Configure Autonomous AI Database for Proof of Concept (POC)

This use case demonstrates how to quickly configure your Autonomous AI Database resources on Dedicated Exadata Infrastructure for developing proof of concept (POC) applications. Oracle Autonomous AI Database on Dedicated Exadata Infrastructure is a highly automated, fully managed database environment running in Oracle Cloud Infrastructure (OCI) with committed hardware and software resources. These isolated resources enable organizations to meet stringent security, availability, and performance requirements while reducing cost and complexity.

If you are looking to quickly create an Autonomous AI Database POC environment, continue to read on.

✓ Tip

For a comprehensive and recommended configuration that involves setting up separate development and production Autonomous AI Database environments, see [Configure Autonomous AI Database with Reference Architecture](#).

Prerequisite Knowledge

To fully understand and appreciate this use case, you should have a basic understanding of Autonomous AI Database on Dedicated Exadata Infrastructure, including its deployment options, key infrastructure components, user roles, and main features. For more detailed information, please refer to [About Autonomous AI Database on Dedicated Exadata Infrastructure](#).

Use Case

Acme Company is considering Autonomous AI Database on Dedicated Exadata Infrastructure for its internal project applications. Before finalizing, Acme I.T. decides to develop a proof of concept (POC) application called **PocApp** using Autonomous AI Database on Dedicated Exadata Infrastructure to help them evaluate the service capabilities.

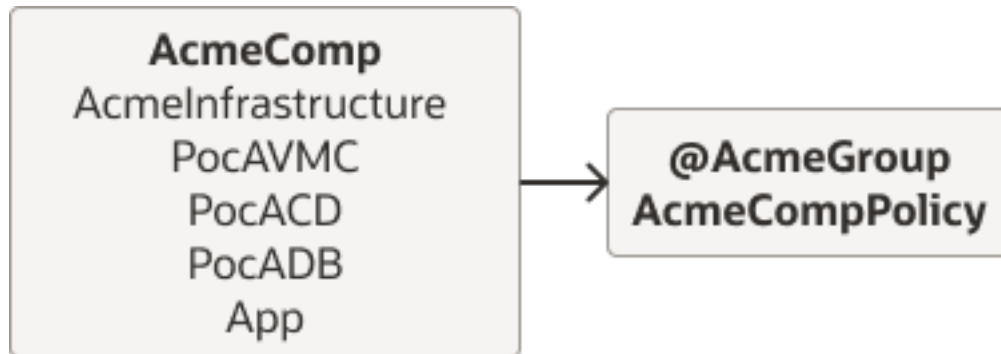
Acme I.T. department will assume the role of fleet administrator, responsible for creating and managing Exadata Infrastructure (EI) and Autonomous Exadata VM Cluster (AVMC) resources for the company. It also takes on the application DBA role to create Autonomous Container Database (ACD) and Autonomous AI Database for their database users.

Note

This example illustrates the fleet administrator creating and managing Autonomous Container Database and Autonomous AI Database resources. However, your organization may prefer that the application DBA undertake this task.

Resources Needed

OCI IAM Components



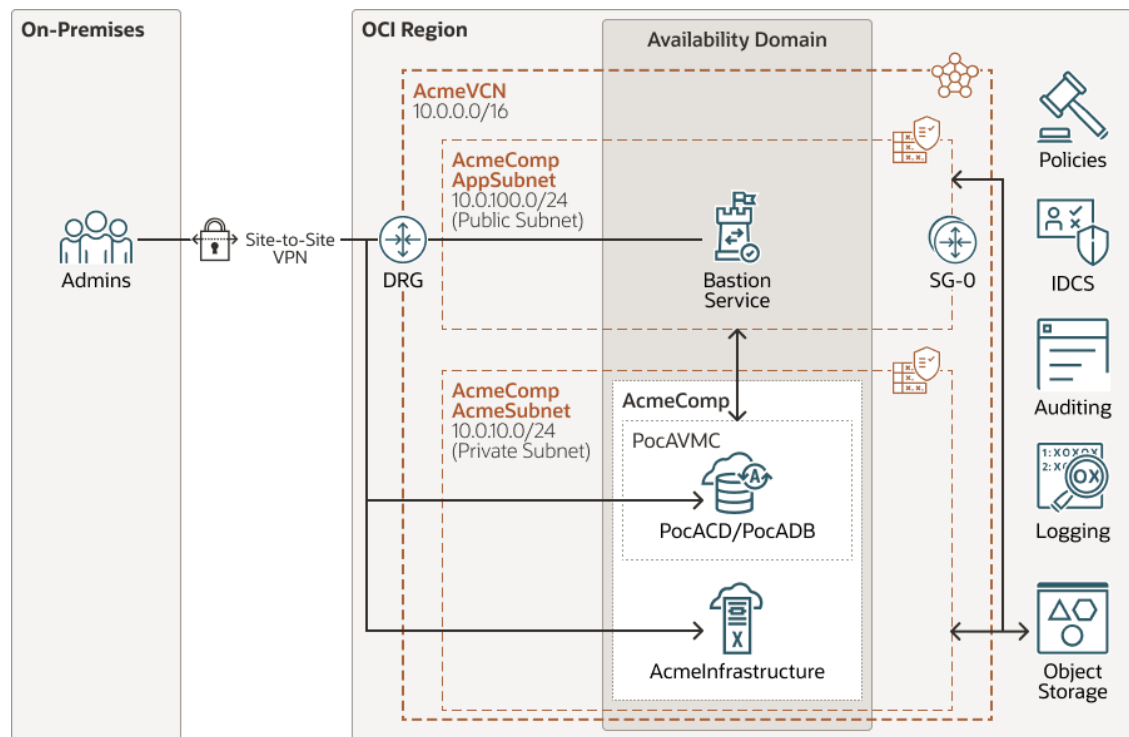
- One compartment named **AcmeComp** to place the resources.
- One group named **AcmeGroup** to which users can be assigned.

Note

Any user added to this group can perform as a fleet administrator, application DBA, or developer.

- One policy called **AcmeCompPolicy** to specify user access to the resources at the compartment and tenancy levels.

Network Resources



represents a security list.

- **Oracle Public Cloud deployments:**

- One VCN to provide network connectivity to all dedicated infrastructure resources. This VCN will connect to the Acme Company VPN using an IPsec VPN and have an Internet Gateway resource that blocks all incoming internet traffic. It will be named **AcmeVCN**.
- Two subnets to provide network access isolation, one for the Autonomous AI Database resources and another for the application's client and mid-tier resources. These subnets will be named **AcmeSubnet** and **AppSubnet** respectively.

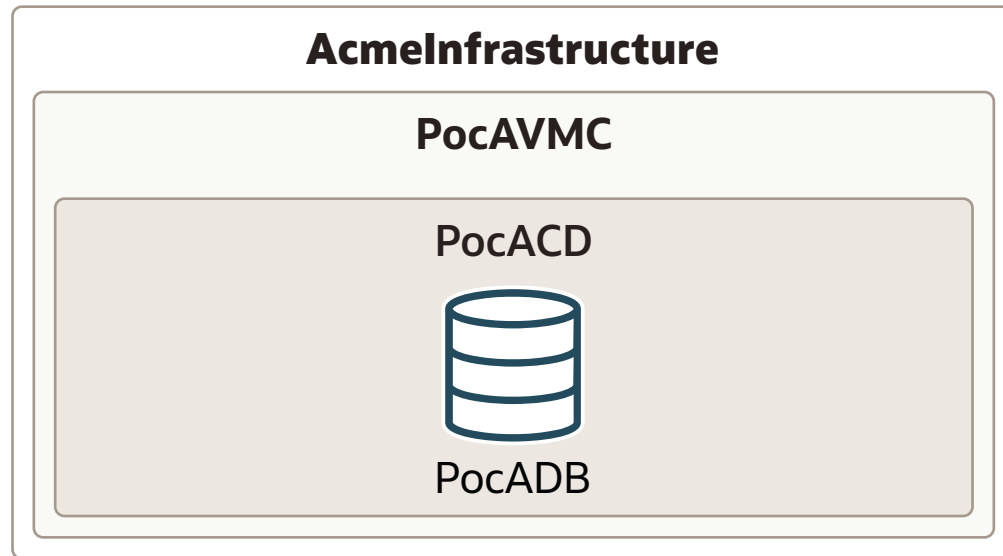
Note

For simplicity, we are using a single VCN and leveraging subnets to provide network isolation. However, you can also create multiple VCNs to provide the required network access isolation. In this example, we create both **AcmeSubnet** and **AppSubnet** under **AcmeComp** for simplicity. Depending on your requirements, you can optionally place these subnets in separate compartments.

- **Exadata Cloud@Customer deployments:**

- Set up network rules as listed in *Network Requirements for Oracle Exadata Database Service on Cloud@Customer* in *Preparing for Exadata Database Service on Cloud@Customer*.
- Additionally, open Port 1522 to allow TCP traffic between primary database and standby database in an Autonomous Data Guard setup.

Autonomous AI Database Resources



Autonomous AI Database resources as per the configuration depicted above.

- One **Exadata Infrastructure** named **AcmeInfrastructure**.
- One Autonomous Exadata VM Cluster (AVMC) within AcmeInfrastructure. This AVMC is named **PocAVMC**.
- PocAVMC hosts the Autonomous Container Database (ACD) and Autonomous AI Database, which are named **PocACD** and **PocADB** respectively, for developing the **PocApp** application.

High-Level Steps

Before Acme I.T. begins configuring Autonomous AI Database resources on Dedicated Exadata Infrastructure, it requests a service limit increase using the OCI console to add Exadata Infrastructure resources - Database Servers and Storage Servers to the tenancy. Refer to Request a Service Limit Increase for more details.

Listed below are the high-level steps to implement this use case:

1. Acme I.T. or security administrator for Acme Company's cloud tenancy creates the **AcmeComp** compartment, **AcmeGroup** group, and **AcmePolicy** compartment policy for resource isolation.
2. For access isolation:
 - For **Oracle Public Cloud** deployments, Acme I.T. or the network administrator for Acme creates the following network resources in the **AcmeComp** compartment:
 - VCN: **AcmeVCN**
 - Private subnets: **AcmeSubnet**
 - Public subnet: **AppSubnet**
 - For **Exadata Cloud@Customer** deployments, Acme I.T. or the network administrator of Acme ensures to:

- Set up network rules as listed in *Network Requirements for Oracle Exadata Database Service on Cloud@Customer* in *Preparing for Exadata Database Service on Cloud@Customer*.
 - Open Port 1522 to allow TCP traffic between primary database and standby database in an Autonomous Data Guard setup.
3. After creating network resources, the security administrator adds the cloud user of a designated Acme I.T. member to the **AcmeGroup**, thus authorizing that user as the fleet administrator.
 4. The newly authorized fleet administrator creates the following dedicated infrastructure resources in the **AcmeComp** compartment, in the below listed order:
 - Exadata Infrastructure resource named **AcmeInfrastructure**.
 - Autonomous Exadata VM Cluster (AVMC) named **PocAVMC**, specifying the newly created Exadata Infrastructure.

Note

For Oracle Public Cloud deployments, use **AcmeVCN** and **AcmeSubnet** as its VCN and subnet.

- Autonomous Container Database (ACD) named **PocACD** in the **AcmeComp** compartment, specifying **PocAVMC** as its underlying resource.
- Autonomous AI Database named **PocADB** in the **AcmeComp** compartment, specifying **PocACD** as its underlying resource.

Step 1. Create OCI IAM Components

In this step, the security administrator for Acme Company's cloud tenancy creates the following OCI IAM components for resource isolation:

- **AcmeComp** compartment.

To perform this step, the security administrator follows the instructions in Managing Compartments in *Oracle Cloud Infrastructure Documentation* to create a compartment using the Oracle Cloud console. When following these instructions, the security administrator specifies the root compartment of the tenancy as the parent compartment of AcmeComp compartment.
- **AcmeGroup** group.

To perform this step, the security administrator follows the instructions in Managing Groups in *Oracle Cloud Infrastructure Documentation* to create a group using the Oracle Cloud console.
- **AcmeCompPolicy** Policy.

To perform this step, the security administrator follows the instructions in Managing Policies in *Oracle Cloud Infrastructure Documentation* to create a policy using the Oracle Cloud console.

Note

In addition to creating the required policy statements, in this example the security administrator also creates "USE tag-namespaces" policy statements to permit group members to assign existing tags to the resources they create. To permit group members to create tags as well as use existing tags, the security administrator would instead create "MANAGE tag-namespaces" policy statements.

When following these instructions to create **AcmeCompPolicy**, the security administrator:

1. Sets the **Compartment** in the side menu to AcmeComp **before** clicking **Create Policy**.
2. Adds either of the following Policy Statements depending on their deployment platform:

— **Oracle Public Cloud deployments:**

- * Allow group AcmeGroup to MANAGE cloud-exadata-infrastructures in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE cloud-autonomous-vmclusters in compartment AcmeComp
- * Allow group AcmeGroup to USE virtual-network-family in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE autonomous-container-databases in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE autonomous-databases in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE autonomous-backups in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE instance-family in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE metrics in compartment AcmeComp
- * Allow group AcmeGroup to INSPECT work-requests in compartment AcmeComp
- * Allow group AcmeGroup to USE tag-namespaces in compartment AcmeComp

— **Exadata Cloud@Customer deployments:**

- * Allow group AcmeGroup to MANAGE exadata-infrastructures in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE autonomous-vmclusters in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE autonomous-container-databases in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE autonomous-databases in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE autonomous-backups in compartment AcmeComp
- * Allow group AcmeGroup to MANAGE instance-family in compartment AcmeComp

- * Allow group AcmeGroup to MANAGE metrics in compartment AcmeComp
- * Allow group AcmeGroup to INSPECT work-requests in compartment AcmeComp
- * Allow group AcmeGroup to USE tag-namespaces in compartment AcmeComp

Step 2. Create the VCN and Subnets

APPLIES TO:  Oracle Public Cloud only

In this step, Acme I.T. or the network administrator of Acme creates the **AcmeVCN** VCN and the **AcmeSubnet** and **AppSubnet** subnets in the **AcmeComp** compartment.

To perform this step, Acme I.T. first confers with the Acme I.T. department's networking to reserve a CIDR IP address range that will not conflict with the company's on-premises network. (Otherwise, the VCN would conflict with the on-premises network and an IPSec VPN could not be set up.) The reserved range is CIDR 10.0.0.0/16.

Then, Acme I.T. adapts the instructions in Scenario B: Private Subnet with a VPN in *Oracle Cloud Infrastructure Documentation* to create the VCN, the Subnets and other network resources using the Oracle Cloud console.

In this example, the following CIDR blocks will be used for the two (2) subnets in **AcmeVCN**:

- 10.0.10.0/24 for **AcmeSubnet** (private subnet)
- 10.0.100.0/24 for **AppSubnet** (public subnet)

When adapting these instructions, Acme I.T. manually creates security lists (instead of using the default security lists) to isolate and separate security rules and thus make network management simpler. These security lists are:

- **AcmeSeclist**: the basic security list for **AcmeSubnet**. It is used when the AcmeSubnet subnet is created.
- **AppSeclist**: the basic security list for **AppSubnet**. It is used when the AppSubnet subnet is created.

For more details on AVMC ingress and egress requirements, see Requirements to Provision an Autonomous Exadata VM Cluster.

Security Rules in AcmeSeclist

Here are the ingress rules created in **AcmeSeclist**:

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.10.0/24	ICMP	All	All	All	ICMP traffic for : All
No	10.0.10.0/24	TCP	All	All		TCP traffic for ports: All
No	10.0.100.0/24	TCP	All	1521		TCP traffic for port: 1521 Oracle Net
No	10.0.100.0/24	TCP	All	2484		TCP traffic for port: 2484 Oracle Net
No	10.0.100.0/24	TCP	All	6200		ONS/FAN for ports: 6200

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.100.0/24	TCP	All	443		HTTPS traffic for port: 443

Here are the egress rules created in **AcmeSecList**:

Stateless	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.10.0/24	ICMP	All	All	All	All ICMP traffic within DevVMSubnet
No	10.0.10.0/24	TCP	All	All		All TCP traffic within DevVMSubnet

Security Rules in AppSecList

Here is the ingress rule created in **AppSecList**:

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	0.0.0.0/0	TCP	All	22	All	SSH traffic for ports: 22

Note

It is recommended to change 0.0.0.0/0 in the security rules to your approved list of CIDR range/IP addresses.

Here are the egress rules created in **AppSecList**:

Stateless	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.10.0/24	TCP	All	1521		
No	10.0.10.0/24	TCP	All	2484		
No	10.0.10.0/24	TCP	All	443		
No	10.0.10.0/24	TCP	All	6200		

Step 3. Assign Fleet Administrator

In this step, the security administrator adds the cloud user of a designated Acme I.T. member to the **AcmeGroup**.

To perform this step, the security administrator follows the instructions in Managing Users in *Oracle Cloud Infrastructure Documentation* to add a user to a group using the Oracle Cloud console.

Step 4. Create Autonomous AI Database Resources

In this step, the fleet administrator creates the following dedicated infrastructure resources in the **AcmeComp** compartment as per the following sequence:

1. Exadata Infrastructure

In this step, the fleet administrator follows the instructions in Create an Exadata Infrastructure Resource to create an Exadata Infrastructure resource named **AcmeInfrastructure**.

2. Autonomous Exadata VM Cluster

In this step, the fleet administrator follows the instructions in Create an Autonomous Exadata VM Cluster to create **PocAVMC** with the following specifications, leaving all the other attributes at their default settings.

Setting	Value
AVMC Name	PocAVMC
Underlying Exadata Infrastructure	AcmeInfrastructure
Virtual cloud network (VCN) APPLIES TO:  Oracle Public Cloud only	AcmeVCN
Subnet APPLIES TO:  Oracle Public Cloud only	AcmeSubnet

3. Autonomous Container Database

In this step, the fleet administrator follows the instructions in Create an Autonomous Container Database to create **PocACD** with the following specifications, leaving all other attributes at their default settings.

Setting	Value
ACD Name	PocACD
Underlying AVMC	PocAVMC
Container Database Software version	Latest software version (N)

4. Autonomous AI Database

In this step, the fleet administrator follows the instructions in Create an Autonomous AI Database to create **PocADB**. These databases are created with the following specifications, leaving all other attributes at their default settings.

Setting	Value
Database Name	PocADB
Underlying ACD	PocACD
Database instance	Can choose to create an Autonomous AI Database or an Autonomous AI Database for Developers

The Autonomous AI Database on Dedicated Exadata Infrastructure is now configured to develop quick proof of concept applications.

Configure Autonomous AI Database with Reference Architecture

This use case demonstrates how to configure your Autonomous AI Database resources on Dedicated Exadata Infrastructure to leverage its capabilities better. This is a comprehensive and recommended configuration that leverages the reference architecture of Autonomous AI Database on Dedicated Exadata Infrastructure.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure is a highly automated, fully managed database environment running in Oracle Cloud Infrastructure (OCI) with committed hardware and software resources. These isolated resources enable organizations to meet stringent security, availability, and performance requirements while reducing cost and complexity.



Tip

If you are looking for guidance to quickly configure an Autonomous AI Database POC environment, see [Configure Autonomous AI Database for Proof of Concept \(POC\)](#).

Prerequisite Knowledge

To fully understand and appreciate this use case, you should have a basic understanding of Autonomous AI Database on Dedicated Exadata Infrastructure, including its deployment options, key infrastructure components, user roles, and main features. For more detailed information, please refer to [About Autonomous AI Database on Dedicated Exadata Infrastructure](#).

Use Case

Acme Company has chosen to use Autonomous AI Database on Dedicated Exadata Infrastructure for its internal project applications. The Acme I.T. department will assume the role of fleet administrator, responsible for creating and managing Exadata Infrastructure (EI) and Autonomous Exadata VM Cluster (AVMC) resources for the company. Within each project team or line of business, designated users will take on the application DBA role, tasked with creating Autonomous Container Database (ACD) and Autonomous AI Databases for their database users, including application developers, testers, and deployers.



Note

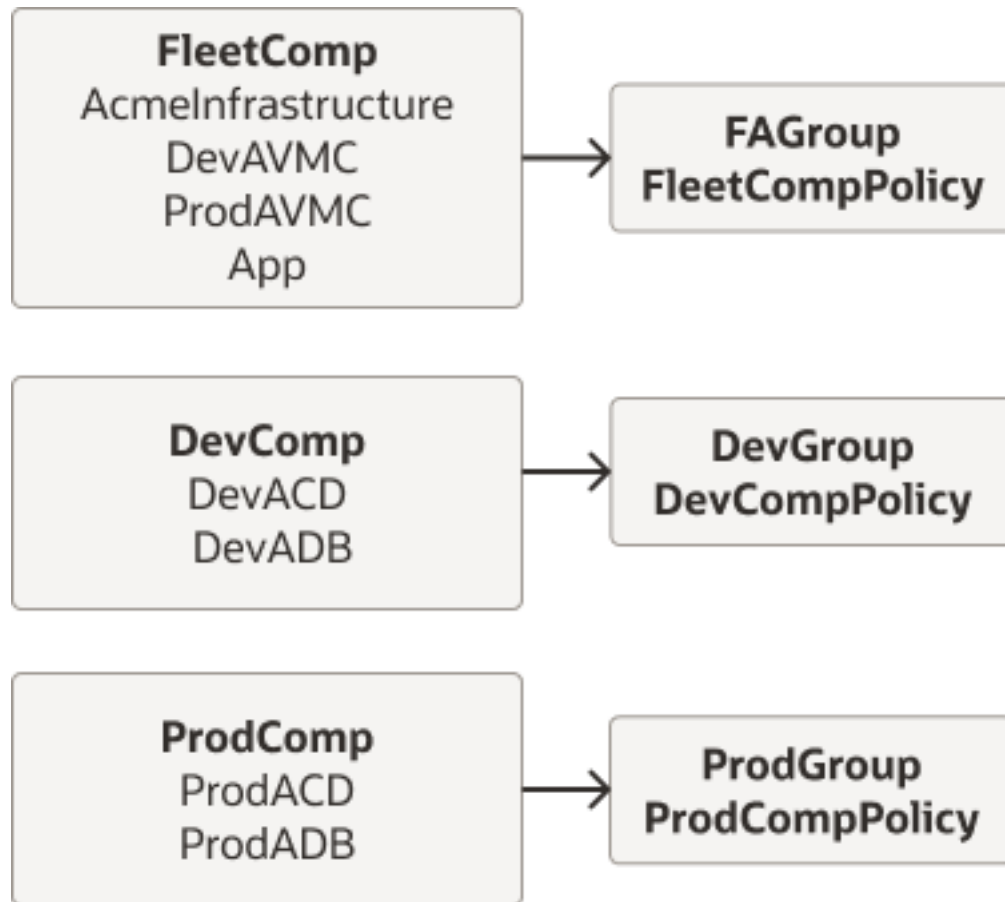
This example illustrates the application DBA creating and managing the ACD resources. However, your organization may prefer that the fleet administrator undertake this task themselves.

Acme I.T. will allocate resources to the various teams, ensuring the provision of AVMCs that meet the required SLAs. Additionally, to control the allocation of resources fairly, Acme I.T. does not want any project team or line of business to have management access to the underlying dedicated infrastructure. Furthermore, in compliance with regulatory audits, Acme management does not want Acme I.T. to be able to access the data that belongs to different project teams or lines of business; that is, the data they store in their application databases.

AcmeHR, an internal HR application developed and utilized by Acme, operates in two distinct environments: one for development and testing (**Dev**) and another for production (**Prod**). Acme I.T. is committed to maintaining strict isolation between these environments to prevent any unauthorized access or interaction between the Dev and Prod teams.

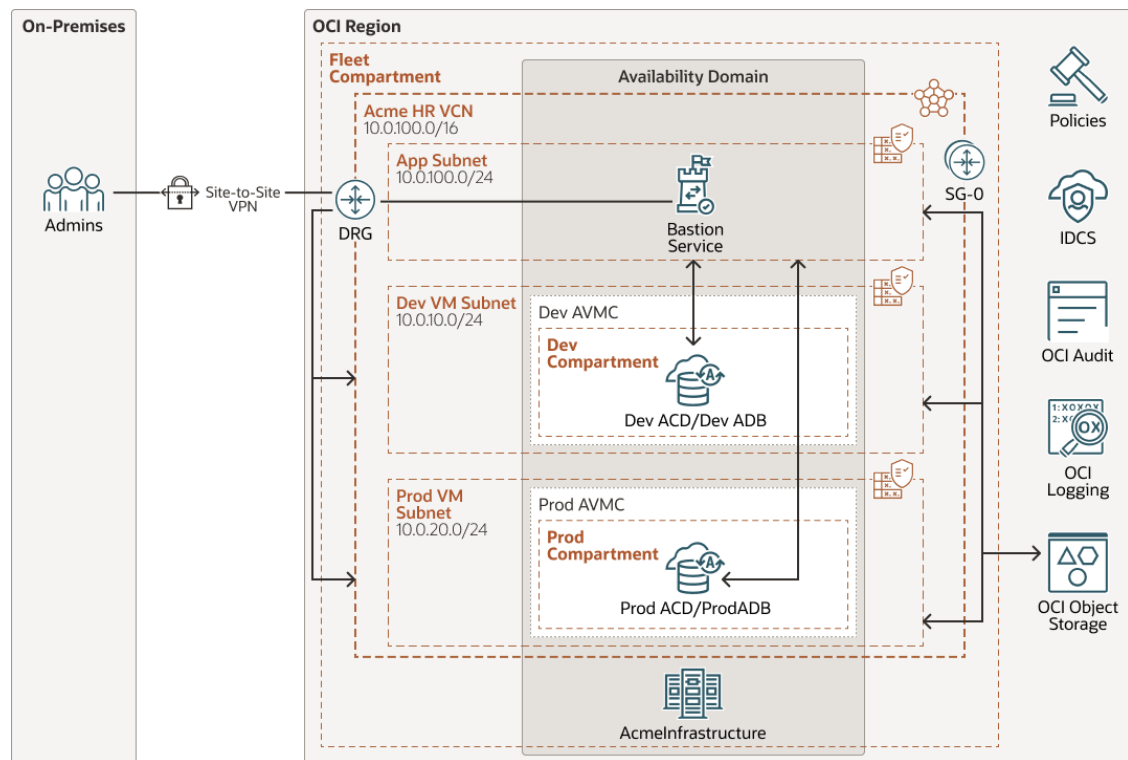
Resources Needed

OCI IAM Components



- Three compartments to provide resource isolation as described below:
 - **FleetComp** for the resources Acme I.T. creates, the networking, and the private subnets used by Dev and Prod databases.
 - **DevComp** for the Dev database.
 - **ProdComp** for the Prod database.
- Three groups to which users can be assigned, one each for Acme I.T., Dev users and Prod users. These groups will be named **FAGroup**, **DevGroup**, and **ProdGroup**.
- Required policies to specify user access to the resources at the compartment and tenancy levels.

Network Resources



represents a security list.

- **Oracle Public Cloud deployments:**

- One VCN to provide network connectivity to all dedicated infrastructure resources. This VCN will connect to the Acme Company VPN using an IPsec VPN and have an Internet Gateway resource that blocks all incoming internet traffic. It will be named **AcmeHRVCN**.
- Three subnets to provide network access isolation, one each for the Autonomous AI Database resources of the **Dev** and **Prod** environments and one for the application's client and mid-tier resources. These subnets will be named **DevVMSubnet**, **ProdVMSubnet**, and **AppSubnet**, respectively.

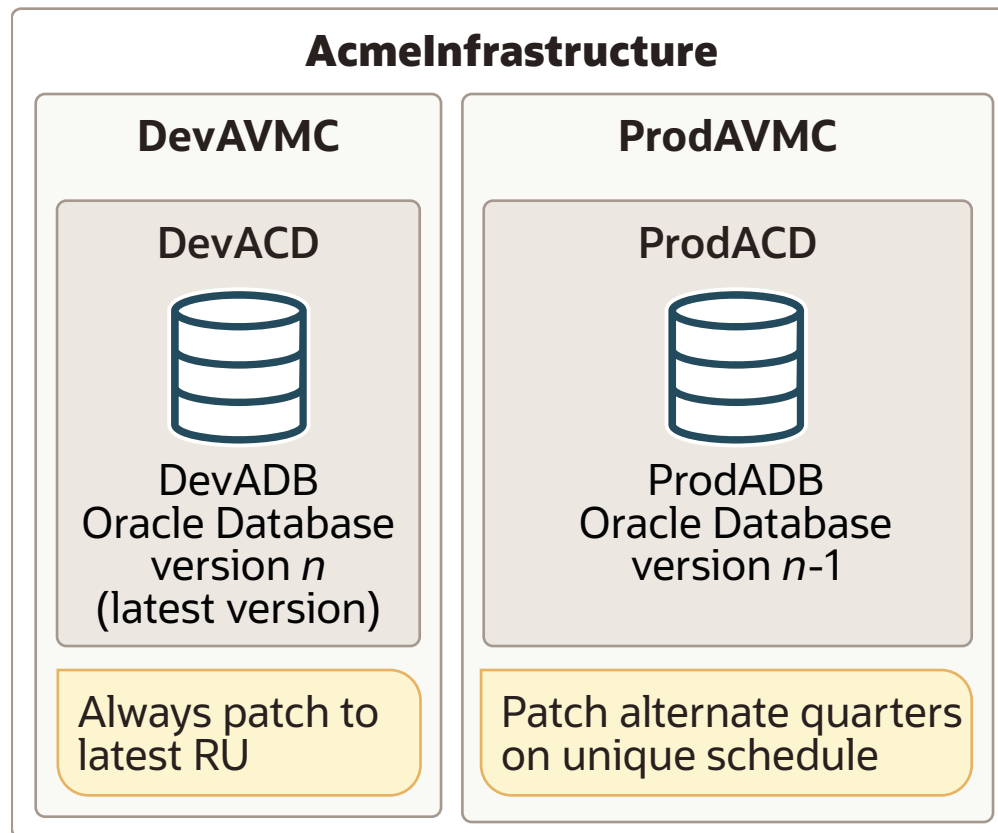
Note

For simplicity, we are using a single VCN and leveraging subnets to provide network isolation. However, you can also create multiple VCNs to provide the required network access isolation. In this example, we create all three subnets: **DevVMSubnet**, **ProdVMSubnet**, and **AppSubnet** under **FleetComp** for simplicity. Depending on your requirements, you can optionally place these subnets in separate compartments.

- **Exadata Cloud@Customer deployments:**

- Set up network rules as listed in *Network Requirements for Oracle Exadata Database Service on Cloud@Customer* in *Preparing for Exadata Database Service on Cloud@Customer*.
- Additionally, open Port 1522 to allow TCP traffic between primary database and standby database in an Autonomous Data Guard setup.

Autonomous AI Database Resources



Autonomous AI Database resources as per the configuration depicted above.

- One **Exadata Infrastructure** to host two AVMCs. It is named **AcmeInfrastructure**.
- Two **AVMCs** within AcmeInfrastructure for the Dev and Prod environments. These AVMCs are named **DevAVMC** and **ProdAVMC**, respectively.
- **DevAVMC:**
 - Hosts the **Autonomous Container Database** and **Autonomous AI Databases**, which are named **DevACD** and **DevADB**, respectively, for developing and testing the AcmeHR application.
 - Always patched to the latest RU (release update).
 - Retains seven (7) days of backups.
 - For every Release Update (RU) or patch release, AcmeHR deployed on DevADB is tested with the latest RU.
 - In case of any critical issues, a one-off patch is requested. After applying the one-off patch, AcmeHR is again validated to ensure that the code is stable and suitable for promotion to production. See Autonomous AI Database Service Maintenance to know more about one-off patches.
- **ProdAVMC:**

- Hosts the **Autonomous Container Database** and **Autonomous AI Database** provisioned for production deployment of the AcmeHR application. They are named **ProdACD** and **ProdADB**, respectively.
- Always runs on the **N-1 software version**.
- Retains backups for a longer duration. If required, long-term backups are also enabled.
- Patched alternate quarters to the validated software, that is, the RUs validated in DevAVMC are deployed in this database.

High-Level Steps

Before Acme I.T. begins configuring Autonomous AI Database resources on Dedicated Exadata Infrastructure, it requests a service limit increase using the OCI console to add Exadata Infrastructure resources - Database Servers and Storage Servers to the tenancy. Refer to Request a Service Limit Increase for more details.

Listed below are the high-level steps to implement this use case:

1. The security administrator for Acme Company's cloud tenancy creates the following compartments, groups, and compartment policies for resource isolation:
 - The **FleetComp**, **DevComp**, and **ProdComp** compartments.
 - The **FAGroup**, **DevGroup**, and **ProdGroup** groups.
 - The **FleetCompPolicy**, **DevCompPolicy**, and **ProdCompPolicy** policies.
2. For access isolation:
 - For **Oracle Public Cloud** deployments, Acme I.T. or the network administrator for Acme creates the following network resources in the **FleetComp** compartment:
 - VCN: **AcmeHRVCN**
 - Private subnets: **DevVMSubnet** and **ProdVMSubnet**
 - Public subnet: **AppSubnet**
 - For **Exadata Cloud@Customer** deployments, Acme I.T. or the network administrator of Acme ensures to:
 - Set up network rules as listed in *Network Requirements for Oracle Exadata Database Service on Cloud@Customer* in Preparing for Exadata Database Service on Cloud@Customer.
 - Open Port 1522 to allow TCP traffic between primary database and standby database in an Autonomous Data Guard setup.
3. After creating network resources, the security administrator adds the cloud user of a designated Acme I.T. member to the **FAGroup**, thus authorizing that user as the fleet administrator.
4. The newly authorized fleet administrator creates the following dedicated infrastructure resources:
 - An Exadata Infrastructure resource **AcmeInfrastructure** in the **FleetComp** compartment.
 - Two Autonomous Exadata VM Clusters (AVMCs) in the **FleetComp** compartment, specifying the newly created Exadata Infrastructure:
 - **DevAVMC**.
For Oracle Public Cloud deployments, specify **AcmeHRVCN** and **DevVMSubnet** as its VCN and subnet.

- **ProdAVMC.**
For Oracle Public Cloud deployments, specify **AcmeHRVCN** and **ProdVMSubnet** as its VCN and subnet.
- 5. The security administrator then adds designated cloud users to the **DevGroup** and **ProdGroup**, thus authorizing them as application DBAs for the **Dev** and **Prod** environments.
- 6. The newly authorized application DBAs create the following resources in their respective work environments, that is **Dev** and **Prod**:
 - Two Autonomous Container Databases (ACDs):
 - **DevACD** in the **DevComp** compartment, specifying **DevAVMC** as its underlying resource
 - **ProdACD** in the **ProdComp** compartment, specifying **Prod AVMC** as its underlying resource.
 - Autonomous AI Database named **DevADB** in the **DevComp** compartment.
 - Autonomous AI Database named **ProdADB** in the **ProdComp** compartment.

Step 1. Create Compartments

In this step, the security administrator for Acme Company's cloud tenancy creates the **FleetComp**, **DevComp**, and **ProdComp** compartments.

To perform this step, the security administrator follows the instructions in Managing Compartments in *Oracle Cloud Infrastructure Documentation* to create a compartment using the Oracle Cloud console. When following these instructions, the security administrator specifies the root compartment of the tenancy as the parent compartment of each of the three compartments.

Step 2. Create Groups

In this step, the security administrator creates the **FAGroup**, **DevGroup**, and **ProdGroup** groups.

To perform this step, the security administrator follows the instructions in Managing Groups in *Oracle Cloud Infrastructure Documentation* to create a group using the Oracle Cloud console.

Step 3. Create Policies

In this step, the security administrator creates the **FleetCompPolicy**, **DevCompPolicy**, and **ProdCompPolicy** policies.

To perform this step, the security administrator follows the instructions in Managing Policies in *Oracle Cloud Infrastructure Documentation* to create a policy using the Oracle Cloud console.

Note

In addition to creating the required policy statements, in this example the security administrator also creates "USE tag-namespaces" policy statements to permit group members to assign existing tags to the resources they create. To permit group members to create tags as well as use existing tags, the security administrator would instead create "MANAGE tag-namespaces" policy statements.

When following these instructions for the **FleetCompPolicy** policy, the security administrator:

1. Sets the **Compartment** in the side menu to FleetComp **before** clicking **Create Policy**.
2. Adds either of the following Policy Statements depending on their deployment platform:
 - **Oracle Public Cloud deployments:**
 - Allow group FAGroup to MANAGE cloud-exadata-infrastructures in compartment FleetComp
 - Allow group FAGroup to MANAGE cloud-autonomous-vmclusters in compartment FleetComp
 - Allow group FAGroup to USE virtual-network-family in compartment FleetComp
 - Allow group FAGroup to USE tag-namespaces in compartment FleetComp
 - Allow group DevGroup to READ cloud-exadata-infrastructures in compartment FleetComp
 - Allow group DevGroup to READ cloud-autonomous-vmclusters in compartment FleetComp
 - Allow group DevGroup to USE virtual-network-family in compartment FleetComp
 - Allow group ProdGroup to READ cloud-exadata-infrastructures in compartment FleetComp
 - Allow group ProdGroup to READ cloud-autonomous-vmclusters in compartment FleetComp
 - Allow group ProdGroup to USE virtual-network-family in compartment FleetComp
 - **Exadata Cloud@Customer deployments:**
 - Allow group FAGroup to MANAGE exadata-infrastructures in compartment FleetComp
 - Allow group FAGroup to MANAGE autonomous-vmclusters in compartment FleetComp
 - Allow group FAGroup to USE tag-namespaces in compartment FleetComp
 - Allow group DevGroup to READ exadata-infrastructures in compartment FleetComp
 - Allow group DevGroup to READ autonomous-vmclusters in compartment FleetComp
 - Allow group ProdGroup to READ exadata-infrastructures in compartment FleetComp
 - Allow group ProdGroup to READ autonomous-vmclusters in compartment FleetComp

When following these instructions for the **DevCompPolicy** policy, the security administrator:

1. Sets the **Compartment** in the side menu to DevComp **before** clicking **Create Policy**.
2. Adds these Policy Statements:
 - Allow group DevGroup to MANAGE autonomous-container-databases in compartment DevComp
 - Allow group DevGroup to MANAGE autonomous-databases in compartment DevComp
 - Allow group DevGroup to MANAGE autonomous-backups in compartment DevComp
 - Allow group DevGroup to MANAGE instance-family in compartment DevComp

- Allow group DevGroup to USE tag-namespaces in compartment DevComp
- Allow group DevGroup to MANAGE metrics in compartment DevComp
- Allow group DevGroup to INSPECT work-requests in compartment DevComp

When following these instructions for the **ProdCompPolicy** policy, the security administrator:

1. Sets the **Compartment** in the side menu to ProdComp **before** clicking **Create Policy**.
2. Adds these Policy Statements:
 - Allow group ProdGroup to MANAGE autonomous-container-databases in compartment ProdComp
 - Allow group ProdGroup to MANAGE autonomous-databases in compartment ProdComp
 - Allow group ProdGroup to MANAGE autonomous-backups in compartment ProdComp
 - Allow group ProdGroup to MANAGE instance-family in compartment ProdComp
 - Allow group ProdGroup to USE tag-namespaces in compartment ProdComp
 - Allow group ProdGroup to MANAGE metrics in compartment ProdComp
 - Allow group ProdGroup to INSPECT work-requests in compartment ProdComp

Step 4. Create the VCN and Subnets

APPLIES TO: • Oracle Public Cloud only

In this step, Acme I.T. or the network administrator of Acme creates the **AcmeHRVCN** VCN and the **DevVMSubnet**, **ProdVMSubnet**, and **AppSubnet** subnets in the **FleetComp** compartment.

To perform this step, Acme I.T. first confers with the Acme I.T. department's networking to reserve a CIDR IP address range that will not conflict with the company's on-premises network. (Otherwise, the VCN would conflict with the on-premises network and an IPSec VPN could not be set up.) The reserved range is CIDR 10.0.0.0/16.

Then, Acme I.T. adapts the instructions in Scenario B: Private Subnet with a VPN in *Oracle Cloud Infrastructure Documentation* to create the VCN, the Subnets and other network resources using the Oracle Cloud console.

In this example, the following CIDR blocks will be used for the three (3) subnets in **AcmeHRVCN**:

- Two private subnets:
 - 10.0.10.0/24 for DevVMSubnet
 - 10.0.20.0/24 for ProdVMSubnet
- One public subnet:
 - 10.0.100.0/24 for AppSubnet

When adapting these instructions, Acme I.T. manually creates security lists (instead of using the default security lists) to isolate and separate security rules and thus make network management simpler. These security lists are:

- DevSecList: the basic security list for **DevVMSubnet**. It is used when the DevVMSubnet subnet is created.
- ProdSecList: the basic security list for **ProdVMSubnet**. It is used when the ProdVMSubnet subnet is created.

- AppSecList: the basic security list for **AppSubnet**. It is used when the AppSubnet subnet is created.

For more details on AVMC ingress and egress requirements, see Requirements to Provision an Autonomous Exadata VM Cluster.

Security Rules in DevSecList

Here are the ingress rules created in **DevSecList**:

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.10.0/24	ICMP	All	All	All	ICMP traffic for : All
No	10.0.10.0/24	TCP	All	All		TCP traffic for ports: All
No	10.0.100.0/24	TCP	All	1521		TCP traffic for port: 1521 Oracle Net
No	10.0.100.0/24	TCP	All	2484		TCP traffic for port: 2484 Oracle Net
No	10.0.100.0/24	TCP	All	6200		ONS/FAN for ports: 6200
No	10.0.100.0/24	TCP	All	443		HTTPS traffic for port: 443

Here are the egress rules created in **DevSecList**:

Stateless	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.10.0/24	ICMP	All	All	All	All ICMP traffic within DevVMSubnet
No	10.0.10.0/24	TCP	All	All		All TCP traffic within DevVMSubnet

Security Rules in ProdSecList

Note

Even though **ProdSecList** has the same security rules as **DevSecList**, the network administrator has created separate security lists instead of creating a single security list for both project teams to accommodate any additional security rules needed by one of the project teams in the future.

Here are the ingress rules created in **ProdSecList**:

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.20.0/24	ICMP	All	All	All	ICMP traffic for : All
No	10.0.20.0/24	TCP	All	All		TCP traffic for ports: All
No	10.0.100.0/24	TCP	All	1521		TCP traffic for port: 1521 Oracle Net
No	10.0.100.0/24	TCP	All	2484		TCPs traffic for port: 2484 Oracle Net
No	10.0.100.0/24	TCP	All	6200		ONS/FAN for ports: 6200
No	10.0.100.0/24	TCP	All	443		HTTPS traffic for port: 443

Here are the egress rules created in **ProdSecList**:

Stateless	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.20.0/24	ICMP	All	All	All	All ICMP traffic within ProdVMSubnet
No	10.0.20.0/24	TCP	All	All		All TCP traffic within ProdVMSubnet

Security Rules in AppSecList

Here is the ingress rule created in **AppSecList**:

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	0.0.0.0/0	TCP	All	22	All	SSH traffic for ports: 22

Note

It is recommended to change 0.0.0.0/0 in the security rules to your approved list of CIDR range/IP addresses.

Here are the egress rules created in **AppSecList**:

Stateless	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.10.0/24	TCP	All	1521		
No	10.0.10.0/24	TCP	All	2484		

Stateless	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows
No	10.0.10.0/24	TCP	All	443		
No	10.0.10.0/24	TCP	All	6200		
No	10.0.20.0/24	TCP	All	1521		
No	10.0.20.0/24	TCP	All	2484		
No	10.0.20.0/24	TCP	All	443		
No	10.0.20.0/24	TCP	All	6200		

Step 5. Assign Fleet Administrator

In this step, the security administrator adds the cloud user of a designated Acme I.T. member to the **FAGroup**.

To perform this step, the security administrator follows the instructions in Managing Users in *Oracle Cloud Infrastructure Documentation* to add a user to a group using the Oracle Cloud console.

Step 6. Create the Exadata Infrastructure Resource

In this step, the fleet administrator follows the instructions in Create an Exadata Infrastructure Resource to create an Exadata Infrastructure resource named **AcmeInfrastructure** in the **FleetComp** compartment.

Step 7. Create the Autonomous Exadata VM Cluster Resources

In this step, the fleet administrator follows the instructions in Create an Autonomous Exadata VM Cluster to create **DevAVMC** and **ProdAVMC** in the **FleetComp** compartment with the following specifications, leaving all the other attributes at their default settings.

Setting	Dev environment	Prod environment
AVMC Name	DevAVMC	ProdAVMC
Underlying Exadata Infrastructure	AcmeInfrastructure	AcmeInfrastructure
Virtual cloud network (VCN) APPLIES TO:  Oracle Public Cloud only	AcmeHRVCN	AcmeHRVCN
Subnet APPLIES TO:  Oracle Public Cloud only	DevVMSubnet	ProdVMSubnet

Step 8. Assign Application DBAs

In this step, the security administrator adds designated cloud users to **DevGroup**, thus authorizing them as application DBAs for the Dev resources, and then repeats the process for **ProdGroup**.

To perform this step, for each user the security administrator follows the instructions in Managing Users in *Oracle Cloud Infrastructure Documentation* to add a user to a group using the Oracle Cloud console.

Step 9. Create Autonomous Container Database Resources

In this step, the respective application DBAs follow the instructions in [Create an Autonomous Container Database](#) to create the Autonomous Container Databases (ACDs) for the **Dev** and **Prod** environments. These ACDs are created with the following specifications, leaving all other attributes at their default settings.

Setting	Dev environment	Prod environment
Compartment	DevComp	ProdComp
ACD Name	DevACD	ProdACD
Underlying AVMC	DevAVMC	ProdAVMC
Container Database Software version	Latest software version (N)	Immediate predecessor of the software version (N-1)
Maintenance version	Latest RU (release update)	Next RU (release update)
Backup retention period	7 days	30 days

Step 10. Create Autonomous AI Database Resources

In this step, the respective application DBAs follow the instructions in [Create an Autonomous AI Database](#) to create the Autonomous AI Databases for the **Dev** and **Prod** environments. These databases are created with the following specifications, leaving all other attributes at their default settings.

Setting	Dev environment	Prod environment
Compartment	DevComp	ProdComp
Database Name	DevADB	ProdADB
Underlying ACD	DevACD	ProdACD
Database instance	Can choose to create an Autonomous AI Database or an Autonomous AI Database for Developers	Autonomous AI Database

The Autonomous AI Database on Dedicated Exadata Infrastructure is now configured to develop and test the **AcmeHR** application, with subsequent deployment in the production environment.

Notifications Example: Emails for Maintenance Events

This example shows how to use the Events and Notifications services to generate emails when maintenance events occur on Exadata Infrastructure and Autonomous Container Database resources. These maintenance events are:

- Maintenance Scheduled for an Exadata Infrastructure resource
- Maintenance Reminder for an Exadata Infrastructure resource
- Maintenance Begin for an Exadata Infrastructure resource
- Maintenance End for an Exadata Infrastructure resource
- Maintenance Scheduled for an Autonomous Container Database resource
- Maintenance Reminder for an Autonomous Container Database resource

- Maintenance Begin for an Autonomous Container Database resource
- Maintenance End for an Autonomous Container Database resource

Use Case

The I.T. department for Acme Company would like emails sent to its Fleet Administrators email group (autonomous-fleet-admin-group@acme.example.com) whenever a maintenance event occurs on any of its Exadata Infrastructure or Autonomous Container Database resources. All of these resources are created in their AcmeContainerDBs compartment.

High-Level Steps

1. Create a notifications service topic.
2. Add an email subscription to the topic.
3. Add an events service rule to send maintenance events to the notifications service topic.

Note

The user who performs these steps must have "manage ons-topics" and "manage ons-subscriptions" policies for the compartment in which the topic and subscription are created; for example:

```
allow group AcmeFAs to manage ons-topics in compartment AcmeContainerDBs
allow group AcmeFAs to manage ons-subscriptions in compartment AcmeContainerDBs
```

Step 1. Create a Notifications Topic

An administrator for Acme Company's cloud tenancy creates the Acme-Autonomous-Maintenance notifications topic in the AcmeContainerDBs compartment.

1. On the Oracle Cloud Infrastructure Home page, click the  menu icon in the top corner to display the side menu, click **Developer Services**, and click **Notifications** under **Application Integration**.

The list of notifications topics in the root compartment is displayed.

2. In the side menu's **Compartment** list, select the **AcmeContainerDBs** compartment.

The list of notifications topics refreshes to show those in the AcmeContainerDBs compartment.

3. Click **Create Topic** to open the Create Topic window.
4. In the Create Topic window, enter **Acme-Autonomous-Maintenance** in the **Name** field and then click **Create**.

The Create Topic window closes and the list of notifications topics refreshes to show the newly created topic.

For detailed information about creating topics, see *Using Console in Oracle Cloud Infrastructure Documentation*.

Step 2. Add a Subscription to the Notifications Topic

An administrator for Acme Company's cloud tenancy adds an email subscription to the Acme-Autonomous-Maintenance notifications topic.

1. Go to the Topic Details page for the Acme-Autonomous-Maintenance notifications topic by clicking its name in the list of notification topics in the AcmeContainerDBs compartment.
2. In the side menu's **Resources** list, make sure **Subscriptions** is selected.
3. Click **Create Subscription** to open the Create Subscription window.
4. In the Create Subscription window, provide the following information:
 - In the **Protocol** list, select **Email**.
 - In the **Email** field, enter **autonomous-fleet-admin-group@acme.example.com**.

When finished, click **Create**.

The Create Subscription window closes and the Subscription Details page for the newly created subscription is displayed. Note that it is **Pending confirmation**.

5. Sign in to an email account that is part of the autonomous-fleet-admin-group group alias. Then, open the email "Oracle Cloud Infrastructure Notifications Service Subscription Confirmation" and confirm the email address.

Step 3. Add an Events Service Rule

An administrator for Acme Company's cloud tenancy creates the Acme-Autonomous-Maintenance events service rule in the AcmeContainerDBs compartment, specifying maintenance events as the triggering condition and delivery to the Acme-Autonomous-Maintenance notifications topic as the action.

1. On the Oracle Cloud Infrastructure Home page, click the  menu icon in the top corner to display the side menu, hover over **Observability & Management** to expand its submenu, and then click **Events Service**.
2. In the side menu's **Compartment** list, select the **AcmeContainerDBs** compartment.
The list of rules shows the events service rules in the AcmeContainerDBs compartment.
3. Click **Create Rule** to open the Create Rule window.
4. In the Create Subscription window, enter **Acme-Autonomous-Maintenance** in the **Display Name** field and complete the **Rule Conditions** and **Actions** sections as follows:
 - **Rule Conditions**

In the first list, select **Event Type** as the basis of the condition. Then, in the **Service Name** list, select **Database**. Finally, in the **Event Type** list, select the following event types one at a time:

- Autonomous Container Database - Maintenance Begin
- Autonomous Container Database - Maintenance End
- Autonomous Container Database - Maintenance Reminder
- Autonomous Container Database - Maintenance Scheduled
- Exadata Infrastructure - Maintenance Begin

- Exadata Infrastructure - Maintenance End
- Exadata Infrastructure - Maintenance Reminder
- Exadata Infrastructure - Maintenance Scheduled

- **Actions**

In the **Action Type** list, select **Notifications**. Then, in the **Notifications Compartment** list, select **AcmeContainerDBs**. Finally, in the **Topic** list, select **Acme-Autonomous-Maintenance**.

When finished, click **Create Rule**.

The Create Subscription window closes and the Details page for the newly created rule is displayed.

6

How-to Guides

Create and Manage Exadata Infrastructure Resources

Create an Exadata Infrastructure Resource

You create an Exadata Infrastructure resource from the **Exadata Infrastructure** page.

Note

If you are a customer with a Multicloud subscription, refer to the instructions in [Create an Exadata Infrastructure in AWS](#) or [Create an Exadata Infrastructure in Azure](#), depending on your Multicloud deployment.

Required IAM Policies

manage cloud-exadata-infrastructures

use vnic

use subnet

Related LiveLabs

For "try it out" alternatives that demonstrate these instructions, see the following labs in Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop:

- Lab 2: Provisioning a Cloud Exadata Infrastructure for Autonomous AI Database on Dedicated Infrastructure
- Lab 4: Provisioning Exadata Infrastructure for Autonomous AI Database on Exadata Cloud@Customer

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous AI Database resource types, click **Exadata Infrastructure**.
The list of Exadata Infrastructure resources in your current Compartment is displayed.
3. Choose the **Compartment** where you want to create an Exadata Infrastructure resource.
The list of Exadata Infrastructure resources refreshes to show those in the selected Compartment.
4. Click **Create Exadata Infrastructure**.

5. On the **Create Exadata Infrastructure** page, enter or select a value for the following:

- **Compartment:** Select a compartment in which you want to create the Exadata Infrastructure resource.
- **Display name:** A user-friendly description or other information that helps you easily identify the resource. The display name does not have to be unique, and you can change it whenever you like.
- **Availability domain:** Select an availability domain where you want the resource placed. This field is applicable for Oracle Public Cloud deployment only.
- **Exadata system model:** Choose the type of Oracle Exadata Database Machine you want to allocate to this resource.

For capacity information about the models and their configurations, see [Characteristics of Infrastructure Shapes](#).

Note

You can also select the **Base System** option to allocate a base system with reduced hardware capacity. Base System supports a single AVM Cluster, similar to X7 and X8 systems. You can not create a Multiple VM - Autonomous VM Cluster (AVMC) on Exadata Infrastructure resources provisioned on an Exadata Base system.

- **Exadata system shape:** Select an Exadata system shape for the Exadata system model chosen above.

Note

X8M-2 and X9M-2 systems on Oracle Public Cloud , X10M systems on Exadata Cloud@Customer, and X11M systems on both Oracle Public Cloud and Exadata Cloud@Customer do not support fixed shapes such as quarter rack, half rack, and full rack.

Note

You can only create a Multiple VM - Autonomous VM Cluster (AVMC) on Exadata Infrastructure resources created in Oracle Cloud after the launch of the multiple VM Autonomous AI Databases feature. Please create a service request in My Oracle Support if you need to address this limitation and want to add an Autonomous VM Cluster to older Exadata Infrastructure resources. For help with filing a support request, see [Create a Service Request in My Oracle Support](#).

- **Compute and Storage Configuration:** Specify the number of database and storage servers for configuring your Exadata Infrastructure resource.
 - **On Oracle Public Cloud:** For X8M-2 and X9M-2 systems, the default configuration is 2 database servers and 3 storage servers. However, you have the flexibility to choose a different value for database servers and storage servers within a valid range. The acceptable range of values is 2 to 32 for database servers and 3 to 64 for storage servers.

Note

As X7-2 and X8-2 systems support fixed shapes such as quarter rack, half rack, and full rack, you can not configure compute and storage for these systems.

- **On Exadata Cloud@Customer deployments:** The number of database servers and storage servers is pre-determined based on the Exadata system shape chosen above. You can not select a different value for database servers, but you can select another value for the Storage servers as per your requirement.

Note

In the case of X10M systems, you can pick the number of database servers and storage servers within a valid range, depending on the X10M shape you chose above.

- **Automatic Maintenance:** Optionally, you can configure the automatic maintenance schedule. On Oracle Public Cloud, click **Edit maintenance preferences**. On Exadata Cloud@Customer, click **Configure Maintenance**. You can then change the maintenance schedule from the default (**No preference**, in Oracle Public Cloud and **Oracle Managed Schedule** in Exadata Cloud@Customer which permits Oracle to schedule maintenance as needed) to **Specify a schedule** in Oracle Public Cloud and **Customer Managed Schedule** in Exadata Cloud@Customer and then selecting the months, weeks, days and hours for the schedule. You can also set a lead time to receive a notification message about an upcoming maintenance from Oracle. For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable.

When you are finished, save your changes.

- **Contact Email(s):** You can provide up to ten contacts' email address for operational notifications and announcements. In Oracle Public Cloud, Click **Add Contact** after entering each email address. In Exadata Cloud@Customer, Provide the Name of the contact, the email address, and an optional phone number. Enable **Secondary maintenance contacts** to add more contacts. Click **+Another Contact** after entering details for one contact.
6. If you want to use Tags, click **Show Advanced Options** and display the **Tags** tab. Then add tags by selecting a **Tag Namespace**, **Tag Key**, and **Tag Value**. Tagging is a metadata system that allows you to organize and track resources within your tenancy. See Tagging Dedicated Autonomous AI Database Cloud Resources.
 7. Optionally, you can save the resource configuration as a stack by clicking **Save as Stack**. You can then use the stack to create the resource through the Resource Manager service.

Enter the following details on the **Save as Stack** dialog, and click **Save**.

- **Name:** Optionally, enter a name for the stack.
- **Description:** Optionally, enter a description for this stack.
- **Save in compartment:** Select a compartment where this Stack will reside.
- **Tag namespace, Tag key, and Tag value:** Optionally, apply tags to the stack.

For requirements and recommendations for Terraform configurations used with Resource Manager, see [Terraform Configurations for Resource Manager](#). To provision the resources defined in your stack, [apply the configuration](#).

8. Submit your details to create the Exadata Infrastructure.

The list of Exadata Infrastructure resources refreshes to show the new resource with a status of **Provisioning** until the new Exadata Infrastructure resource is available.

Note

When you create an Exadata Infrastructure resource, infrastructure billing is initiated, billed by the second with a minimum usage period 48 hours.

Manage Exadata Infrastructure

The Exadata Infrastructure (EI) resource represents the Exadata Database Machine system, together with the networking configuration that connects it to Oracle Cloud. It hosts the underlying components Autonomous VM Clusters, Autonomous Container Databases, and Autonomous AI Databases.

You can use the following sets of instructions to manage your Exadata Infrastructure from the OCI console.

View a List of Exadata Infrastructure Resources

You view a list of Exadata Infrastructure resources by going to the **Exadata Infrastructure** page.

Required IAM Policies

`inspect cloud-exadata-infrastructures`

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous AI Database resource types, click **Exadata Infrastructure**.

The Exadata Infrastructure page is displayed, showing the list of Exadata Infrastructure resources in your current compartment. Choose the **Compartment** where you want to see the list of Exadata Infrastructure resources.

Features of the Exadata Infrastructure Page

You can perform a variety of actions on the Exadata Infrastructure page.

Action	Steps
Create an Exadata Infrastructure resource	Click Create Exadata Infrastructure and follow the instructions in Create an Exadata Infrastructure Resource.
View details of a listed Exadata Infrastructure resource	In the list of Exadata Infrastructure resources, click the name of the resource whose details you want to view.

Action	Steps
Copy the OCID of a listed Exadata Infrastructure resource	In the list of Exadata Infrastructure resources, click the menu in the row of the resource whose OCID you want to copy and then click Copy OCID . The OCID is copied to your clipboard and the menu item changes to Copied.
Apply tags to a listed Exadata Infrastructure resource	In the list of Exadata Infrastructure resources, click the menu in the row of the resource you want to apply tags to and then click Manage tags .
Move the Exadata Infrastructure resource to a different compartment	In the list of Exadata Infrastructure resources, click the menu in the row of the resource you want to move and then click Move resource . Choose the new compartment and submit your changes.
Terminate a listed Exadata Infrastructure resource	In the list of Exadata Infrastructure resources, click the menu in the row of the resource you want to terminate the resource. On Oracle Public Cloud, click Terminate . On Exadata Cloud@Customer, click Delete . Then, in the subsequent dialog: • Read the warning about terminating an Exadata Infrastructure resource. • Enter the name of the Exadata Infrastructure resource. • Submit your details to begin the termination process.

View Details of an Exadata Infrastructure Resource

You view the details of an Exadata Infrastructure resource by going to its **Details** page.

Required IAM Policies

`inspect cloud-exadata-infrastructures`

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous AI Database resource types, click **Exadata Infrastructure**.
The list of Exadata Infrastructure resources in your current Compartment is displayed.
3. Choose the **Compartment** where you want to see the list of Exadata Infrastructure resources.
The list of Exadata Infrastructure resources refreshes to show those in the selected Compartment.
4. Click the name of the Exadata Infrastructure resource whose details you want to view.
The **Details** page for the chosen Exadata Infrastructure resource is displayed.

Features of the Details Page

You can perform a variety of actions on the **Details** page for an Exadata Infrastructure resource.

Action	Steps
Copy the resource's OCID	In the General Information section, click Copy in the OCID field.
Edit the resource's maintenance schedule	Under Actions , click Edit maintenance preferences .
 Note You have another way to edit the maintenance schedule. On Oracle Public Cloud, click Maintenance, and from the <i>Maintenance</i> page, click Edit maintenance run. On Exadata Cloud@Customer, click Maintenance scheduling plan, and from the <i>Maintenance Scheduling Plan</i> page, click Edit Maintenance Preferences.	
View upcoming and past maintenance	Click Maintenance . A list of upcoming planned maintenance activities, if any, is displayed. To view past maintenance activities, click Maintenance History .
Scale up the resource	Click Scale Infrastructure and enter new values for Database Servers and/or Storage Servers choosing the appropriate option. Submit your changes. You can only enter a value that is greater than the current value of the database or storage servers for the Exadata Infrastructure resource.
Add tags to the resource	Click Tags , and from the Tags page, click Add .
Move the resource to a different compartment	Under Actions , click Move resource .
Terminate the resource	On Oracle Public Cloud, click Terminate under Actions . On Exadata Cloud@Customer, click Delete under Actions . Follow the instructions given in Terminate an Exadata Infrastructure Resource .
Update the display name	APPLIES TO:  Oracle Public Cloud only Click Update Display Name under Actions . Enter the new and current display names, and submit your changes.
View the tags applied to the resource	Click the Tags tab.

Click **Autonomous Exadata VM clusters** to view a list of Autonomous Exadata VM Clusters in the selected Compartment that use the chosen Exadata Infrastructure resource. Using this list you can perform a variety of actions.

Action	Steps
Create an Autonomous Exadata VM Cluster	Click Create Autonomous Exadata VM Cluster and follow the instructions in Create an Autonomous Exadata VM Cluster.
View details of a listed Autonomous Exadata VM Cluster	In the list of Autonomous Exadata VM Cluster(s), click the name of the one whose details you want to view.
Move a listed Autonomous Exadata VM Cluster to a different compartment	In the list of Autonomous Exadata VM Cluster(s), click the  menu in the row where you want to move the resource to a different compartment, and click Move resource .
Add tags to a listed Autonomous Exadata VM Cluster	In the list of Autonomous Exadata VM Cluster(s), click the  menu in the row where you want to apply tags to, and click Manage tags .
Copy the OCID of a listed Autonomous Exadata VM Cluster	In the list of Autonomous Exadata VM Cluster(s), click the  menu in the row of the one whose OCID you want to copy and then click Copy OCID . The OCID is copied to your clipboard and the menu item changes to Copied.
Add security attributes	APPLIES TO:  Oracle Public Cloud only In the list of Autonomous Exadata VM Cluster(s), click the  menu in the row of the one you want to add security attributes, click Add security attributes .
Terminate a listed Autonomous Exadata VM Cluster	In the list of Autonomous Exadata VM Cluster(s), click the  menu in the row of the one you want to terminate and then click Terminate . Then, follow the instructions given in Terminate an Exadata Infrastructure Resource .

Move an Exadata Infrastructure Resource to a Different Compartment

You can move an Exadata Infrastructure resource to a different Oracle Cloud Infrastructure compartment.

APPLIES TO:  Oracle Public Cloud APPLIES TO:  Exadata Cloud@Customer

Note

As soon as you move an Exadata Infrastructure resource to a different compartment, the policies that govern the new compartment apply immediately and affect access to the resource. Therefore, both your and other Oracle Cloud users' access to the resource may change, depending on the policies governing the user account's access to resources. For example, a user may lose the ability to create Autonomous Exadata VM Clusters on the resource, given its new compartment.

Required IAM Policies

use `cloud-exadata-infrastructures` in the resource's current compartment and in the compartment you are moving it to.

Procedure

1. Go to the **Details** page of the Exadata Infrastructure resource whose maintenance schedule you want to change.
For instructions, see [View Details of an Exadata Infrastructure Resource](#).
2. Under **Actions**, click **Move resource**.
3. In the **Move resource to a Different Compartment** page, select the new compartment.
4. Click **Move resource**.

Manage Customer Contacts for an Exadata Infrastructure

You can add, edit, and remove customer contacts for an Exadata Infrastructure from its **Details** page.

Required IAM Policies

use `cloud-exadata-infrastructures`

Procedure

1. Go to the **Details** page of the Exadata Infrastructure for which you want manage customer contacts. For instructions, see [View Details of an Exadata Infrastructure Resource](#).
2. On Oracle Public Cloud, click **Manage** next to the **Customer contact** details in the **Maintenance** section. On Exadata Cloud@Customer, click **Manage Contacts** under **Actions**. The page to manage contacts opens, enabling you to perform the following actions:

Add Customer Contacts

You can provide up to ten contacts' email address for operational notifications and announcements. In Oracle Public Cloud, Click **Add Contact** after entering each email address. In Exadata Cloud@Customer, Provide the Name of the contact, the email address, and an optional phone number. Enable **Secondary maintenance contacts** to add more contacts. Click **+Another Contact** after entering details for one contact.

Edit customer contacts

In Oracle Public Cloud, Select the email addresses that you are editing. Click **Save** after making the changes. In Exadata Cloud@Customer, click on the field you want to make changes (name or email address or phone number). You can modify the details of a primary contact or secondary contacts. Save your changes.

Delete customer contacts

Select the contact detail to be removed. Click **X** at the end of the row. The contact details are removed. Save your changes.

Terminate an Exadata Infrastructure Resource

You terminate an Exadata Infrastructure resource by clicking **Terminate** on its **Details** page.

Required IAM Policies

manage cloud-exadata-infrastructures

use vnic

use subnet

Procedure

1. Go to the **Details** page of the Exadata Infrastructure resource which you want to terminate.
For instructions, see View Details of an Exadata Infrastructure Resource.
2. On Oracle Public Cloud, click **Terminate** under **Actions**. On Exadata Cloud@Customer, click **Delete** under **Actions**. In the corresponding dialog:
 - Read the warning about terminating the Exadata Infrastructure resource.
 - Enter the name of the Exadata Infrastructure resource.
 - Confirm your action to begin the termination process.

Manage Exadata Infrastructure Maintenance

Oracle schedules and performs all patching and other maintenance operations on all the infrastructure resources of an Autonomous AI Database. At the same time, it also provides you with various options to customize, view, and reschedule maintenance events for the different infrastructure resources, including Exadata Infrastructure resources.

View and Manage Scheduled Maintenance of an Exadata Infrastructure Resource

You view and manage scheduled maintenance of an Exadata Infrastructure resource from its **Details** page.

Required IAM Policies

use cloud-exadata-infrastructures

Procedure

1. Go to the **Details** page of the Exadata Infrastructure resource whose maintenance schedule you want to change.
For instructions, see View Details of an Exadata Infrastructure Resource.
2. Click **Maintenance**.
The **Exadata Infrastructure Maintenance** page lists any upcoming planned maintenance.
For more information about the details displayed in the **Exadata Infrastructure Maintenance** page, see Scheduled Maintenance Details.
3. For each maintenance event listed in the **Exadata Infrastructure Maintenance** page, you can perform the management operations listed in Management Operations on a Scheduled Maintenance, provided that the event is not already in progress.

When you reschedule an Exadata Infrastructure maintenance event that is already scheduled, Oracle may place it in a queue if its Autonomous Exadata VM Cluster or Autonomous Container Databases are:

- Already undergoing a maintenance update, or
- Scheduled for a maintenance activity simultaneously as the Exadata Infrastructure.

For more information about automatic queuing of maintenance events, see [Automatic Queuing of Maintenance Events](#).

Change the Maintenance Schedule of an Exadata Infrastructure Resource

You view the details of an Exadata Infrastructure resource by going to its **Details** page.

Required IAM Policies

use `cloud-exadata-infrastructures`

Procedure

To change the maintenance schedule of an Exadata Infrastructure:

1. Go to the **Details** page of the Exadata Infrastructure resource whose maintenance schedule you want to change.
For instructions, see [View Details of an Exadata Infrastructure Resource](#).
2. You can change the existing maintenance schedule of the Exadata Infrastructure resource. Under Actions, click **Edit maintenance preferences**.

Note

You have another way to edit maintenance schedule. On Oracle Public Cloud, click **Maintenance** and from the *Maintenance* page, click **Edit maintenance run**. On Exadata Cloud@Customer, click **Maintenance scheduling plan** and from the *Maintenance Scheduling Plan* page, click **Edit Maintenance Preferences**.

3. Change the settings for the Maintenance method and Maintenance schedule as desired.
 - **Configure Maintenance schedule:** Choose one of the following options.
 - Select **No preference** to permit Oracle to schedule maintenance as needed. In Exadata Cloud@Customer this choice would be **Oracle managed schedule**.
 - Select **Specify a schedule** to restrict when Oracle can schedule maintenance. You can choose your preferred month, week, weekday, start time, and lead time for infrastructure maintenance. In Exadata Cloud@Customer this choice would be **Customer managed schedule**.

Tip

For guidance on choosing a custom schedule, see [Settings in Maintenance Schedule](#) that are Customizable.

- **Configure maintenance method:** By default, Exadata Infrastructure is updated in a rolling fashion, one server at a time. You can select non-rolling maintenance to update database and storage servers in parallel. However, non-rolling maintenance incurs a full system downtime.

4. When finished, save your changes.

View Past Maintenance of an Exadata Infrastructure Resource

You view past maintenance of an Exadata Infrastructure resource from its **Details** page.

Required IAM Policies

`inspect cloud-exadata-infrastructures`

Procedure

1. Go to the **Details** page of the Exadata Infrastructure resource whose maintenance schedule you want to change.
For instructions, see [View Details of an Exadata Infrastructure Resource](#).
2. Click **Maintenance History** to see a list of past maintenance.

Create and Manage Autonomous Exadata VM Clusters

Create Autonomous Exadata VM Cluster

You can create an Autonomous Exadata VM cluster from the Oracle Cloud Infrastructure console.

Note

If you are a customer with a Multicloud subscription, refer to the instructions in [Create an Autonomous VM Cluster \(AVMC\) in AWS](#) or [Create an Autonomous VM Cluster \(AVMC\) in Azure](#), depending on your Multicloud deployment.

IAM Policy Requirements

Deployment Choice	IAM Policies
Oracle Public Cloud and Multicloud	<code>manage cloud-autonomous-vmclusters</code> <code>use cloud-exadata-infrastructures</code>
Exadata Cloud@Customer	<code>manage autonomous-vmclusters</code> <code>use exadata-infrastructures</code>

Zero Trust Packet Routing (ZPR) Policy Requirements

APPLIES TO:  Oracle Public Cloud only

1. IAM policies required to add Zero Trust Packet Routing (ZPR) security attributes while provisioning an AVMC:

```
allow group <group_name> to { ZPR_TAG_NAMESPACE_USE,  
SECURITY_ATTRIBUTE_NAMESPACE_USE } in tenancy  
allow group <group_name> to manage autonomous-database-family in tenancy  
allow group <group_name> to read security-attribute-namespaces in tenancy
```

2. To provision an AVMC with security attributes, you must also enable the corresponding ZPR policies to allow traffic.

For example, you must define the sample ZPR policies shown below to provision an AVMC with the following security attributes:

- VCN network: ADBD
- Database client: ADBDClient
- Database server: ADBDServer

Sample ZPR Policies to provision AVMC successfully:

```
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Server:ADBDServer endpoints with protocol='tcp/2484'
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Server:ADBDServer endpoints with protocol='tcp/22'
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Server:ADBDServer endpoints with protocol='tcp/1521'
in VCN-Network:ADBD VCN allow Db-Server:ADBDServer endpoints to connect to
all-endpoints
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Server:ADBDServer endpoints with protocol='icmp'
in VCN-Network:ADBD VCN allow Db-Server:ADBDServer endpoints to connect to
'osn-services-ip-addresses' with protocol='tcp/443'
```

Sample ZPR Policies for successful customer connectivity:

```
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Client:ADBDClient endpoints with protocol='tcp/22'
in VCN-Network:ADBD VCN allow Db-Client:ADBDClient endpoints to connect to
Db-Server:ADBDServer endpoints with protocol='tcp/1521'
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Server:ADBDServer endpoints with protocol='tcp/2484'
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Server:ADBDServer endpoints with protocol='tcp/22'
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Server:ADBDServer endpoints with protocol='tcp/1521'
in VCN-Network:ADBD VCN allow Db-Server:ADBDServer endpoints to connect to
all-endpoints
in VCN-Network:ADBD VCN allow all-endpoints to connect to Db-
Server:ADBDServer endpoints with protocol='icmp'
in VCN-Network:ADBD VCN allow Db-Server:ADBDServer endpoints to connect to
'osn-services-ip-addresses' with protocol='tcp/443'
```

Resource Requirements

To create *one Autonomous Exadata VM Cluster*, you need **at least**:

- 40 ECPU's per node
- 120GB memory per node
- 338.5GB local storage per node
- 6.61TB Exadata storage

Network Requirements

In Oracle Public Cloud deployments, you can provision new VM Clusters with IPv4/IPv6 dual-stack networking, enabling both IPv4 and IPv6 addresses. This option is enabled while you create a VCN, and while creating a subnet.

In Exadata Cloud@Customer deployments, if you have enabled a Disaster Recovery Network, you can use the 3rd NIC that you have configured for specific database operations like Autonomous Data Guard (AuDG) and cloning the Autonomous AI Database. To use the 3rd NIC with your Autonomous Virtual Machine Cluster (AVMC), you must first submit a Service Request (SR) to Oracle Support. This Service Request must be made before the AVMC is created. For help with filing a support request, see [Create a Service Request in My Oracle Support](#). See [Network Requirements for Oracle Exadata Database Service on Cloud@Customer](#) for more details on the network requirements.

Subnet Sizes

- Oracle Public Cloud and Multicloud Deployments:**

If you are provisioning an Autonomous Exadata VM Cluster resource and setting up disaster recovery with Autonomous Data Guard, make sure the IP address space of the Virtual Cloud Networks (VCNs) does not overlap.

The following table lists the minimum required subnet sizes for Autonomous AI Database deployments on Oracle Public Cloud.

✓ **Tip**

The Networking service [reserves three IP addresses in each subnet](#). Allocating a larger space for the subnet than the minimum required (for example, at least /25 instead of /28) can reduce the relative impact of those reserved addresses on the subnet's available space.

Rack Size	Client Subnet: # Required IP Addresses	Client Subnet: Minimum Size
Base System or Quarter Rack	(4 addresses * 2 nodes) + 3 for SCANs + 3 reserved in subnet = 14	/28 (16 IP addresses)
Half Rack	(4 * 4 nodes) + 3 + 3 = 22	/27 (32 IP addresses)
Full Rack	(4* 8 nodes) + 3 + 3 = 38	/26 (64 IP addresses)
Flexible infrastructure systems (X8M and higher)	4 addresses per database node (minimum of 2 nodes) + 3 for SCANs + 3 reserved in subnet	/28 is the minimum subnet size, but the addition of each node increases the subnet size correspondingly.

✓ **Tip**

Ensure you configure your subnets to have sufficient available IP addresses to create multiple Autonomous Exadata VM Cluster. For example, the minimum required IP address for a Base system or Quarter Rack for a single VM Cluster is 14. If you plan to create 2 VM clusters, define your subnet at /27 or have two subnets at /28 level.

- Exadata Cloud@Customer Deployments:**

For Exadata Cloud@Customer IP requirements, refer to IP Addresses and Subnets for Exadata Cloud at Customer.

Security Lists

- **Oracle Public Cloud and Multicloud Deployments:**

- To meet the minimum Security List requirements for provisioning an Autonomous VM Cluster:
 - * Open TCP on all ports for ingress and egress for the subnet CIDR.
 - * Open all ICMP types and codes for ingress and egress for the subnet CIDR.
 - * Ensure that ingress and egress have same type: either both are Stateless or both are Stateful.

For example, in the screenshot below, TestVCN has an **IPv4 CIDR Block** of 11.0.0.0/16 and FleetSubnet (the subnet used to provision AVMC) has an **IPv4 CIDR Block** of 11.0.0.0/24.

The screenshot displays the Oracle Cloud console interface for a Virtual Cloud Network (VCN). The top section, 'VCN Information', shows details for a VCN named 'TestVCN' in the 'FleetCompartment'.

Property	Value
Compartment	FleetCompartment
Created	Apr 29, 2021, 00:55 UTC
IPv4 CIDR Block	11.0.0.0/16
IPv6 Prefix	—
OCID	ocid1.vcn.oc4..
DNS Resolver	JavaSDK.DBVCN.1619657730999
Default Route Table	Default Route Table for JavaSDK.DBVCN.1619657730999
DNS Domain Name	javasdk504.oraclevm.com

The bottom section shows 'Ingress Rules' and 'Egress Rules' for the VCN. The 'Ingress Rules' table lists four rules:

Stateless	Source	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
☐	No	0.0.0.0/0	TCP	All	22	TCP traffic for ports 22 SSH Remote Login Protocol	...
☐	No	0.0.0.0/0	ICMP		3, 4	ICMP traffic for: 3, 4 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set	...
☐	No	11.0.0.0/16	ICMP		3	ICMP traffic for: 3 Destination Unreachable	...
☐	No	11.0.0.0/16	All Protocols			All traffic for all ports	...

The 'Egress Rules' section shows two rules:

Stateless	Destination	IP Protocol	Source Port Range	Destination Port Range	Type and Code	Allows	Description
☐	No	0.0.0.0/0	All Protocols			All traffic for all ports	...
☐	No	0.0.0.0/0	All Protocols			All traffic for all ports	...

Note

In Oracle Public Cloud deployments, when you configure the security rule in the specific subnet where you are provisioning the AVMC, choose source CIDR: as **::/64** for **ipv6** addresses and choose **Select ipv6-ICMP** in the **IP Protocol**. You need the **ipv6-ICMP** protocol between the nodes in ingress and egress, otherwise ping between the nodes does not work and any database operations or patching operations fail with the error *Node not reachable via ICMP ping*.

- Optionally, depending on your requirement, you may also need to open the following ports to connect and operate Autonomous AI Databases created in this AVMC.

Destination Port	IP Protocol	Purpose
1521	TCP	SQL*Net traffic
2484	TCP	Transport Layer Security (TLS) from the Application Subnets
6200	TCP	Oracle Notification Service (ONS) traffic Fast Application Notification (FAN) traffic
443	TCP	Oracle Application Express Oracle Database Actions

- Exadata Cloud@Customer Deployments:**

- Before provisioning an AVMC on Exadata Cloud@Customer, you must setup the network rules as listed in *Network Requirements for Oracle Exadata Database Service on Cloud@Customer* in Preparing for Exadata Database Service on Cloud@Customer.
- Additionally, you need to open *Port 1522* to :
 - * Allow TCP traffic between primary and standby databases in an Autonomous Data Guard setup.
 - * Support database cloning.
- Optionally, if you plan to connect to the following commonly used applications, you may also need to open the following ports.

Destination Port	IP Protocol	Purpose
443 (Egress)	TCP	Key Management Service (KMS)
1521 (Ingress)	TCP	Oracle Enterprise Manager (OEM)
1521	TCP	SQL*Net traffic
2484	TCP	Transport Layer Security (TLS) from the Application Subnets
6200	TCP	Oracle Notification Service (ONS) traffic Fast Application Notification (FAN) traffic
443	TCP	Oracle Application Express Oracle Database Actions

Note

You can only create a Multiple VM - Autonomous VM Cluster (AVMC) on Exadata Infrastructure resources created in Oracle Cloud after the launch of the multiple VM Autonomous AI Databases feature. Please create a service request in My Oracle Support if you need to address this limitation and want to add an Autonomous VM Cluster to older Exadata Infrastructure resources. For help with filing a support request, see [Create a Service Request in My Oracle Support](#).

Instructions

Tip

For "try it out" alternatives that demonstrate these instructions, see the following labs in Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop:

- Lab 3: Provisioning a Cloud Autonomous Exadata VM Cluster for Autonomous AI Database on Dedicated Infrastructure
- Lab 5: Provisioning an Autonomous VM Cluster on Exadata Cloud@Customer

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see [Access Autonomous AI Database in the Oracle Cloud Infrastructure Console](#).
2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous Exadata VM Clusters**.
You can see the list of Autonomous Exadata VM Clusters in your current Compartment.
3. Choose the **Compartment** where you want to create an Autonomous Exadata VM Cluster.
The list of Autonomous Exadata VM Clusters refreshes to show those in the selected Compartment.
4. Click **Create Autonomous Exadata VM Cluster**.
5. Fill out the Create Autonomous Exadata VM Cluster page with the following information:

Setting	Description	Notes
Basic Information	Compartment: Select a compartment to host the Autonomous Exadata VM Cluster. Display Name: Enter a user-friendly description or other information that helps you easily identify the infrastructure resource.	The display name does not have to be unique. Avoid entering confidential information.
Exadata Infrastructure	The Exadata Infrastructure to host the new Autonomous Exadata VM Cluster.	-
Autonomous VM Cluster resources: Compute Model	Compute Model: Denotes the compute model for your Autonomous Exadata VM Cluster resource. Default model is ECPU. This is based on the number of cores elastically allocated from a pool of compute and storage servers.	Starting May 28th, 2025, Autonomous AI Database does not allow you to create new AVMCs using the OCPU billing metric. All the new AVMCs can only be provisioned with ECPUs. Existing OCPU AVMCs and Autonomous AI Databases will continue to operate as usual. You can have your OCPU AVMCs and respective Autonomous AI Databases updated to ECPU via service requests. See Doc ID 2998755.1, for more information.

Setting	Description	Notes
Autonomous VM Cluster resources: DB Server Selection	Lists the DB Servers (VMs) used to deploy the new Autonomous Exadata VM Cluster (AVMC) resource. The maximum resources (CPUs, Memory, and Local Storage) available per VM is also displayed. Optionally, you can add or remove the VMs by clicking Edit DB Server Selection. Clicking this button launches the Change DB Servers dialog listing all the available DB Servers with the following details for each one of them: • Available CPUs • Available memory (GB) • Available local storage (GB) • Number of VM Clusters and Autonomous VM Clusters in it. By default, all the DB Servers that have the minimum resources needed to create an Autonomous VM Cluster are selected. You can remove a	You must select at least 2 DB Servers to deploy an AVMC resource. A minimum of 2 DB Servers are needed for high availability (HA) configuration. Note You can not add or remove DB Servers after provisioning an AVMC.

Setting	Description	Notes
	DB Server by clicking the check-box against it in the list. Depending on your selection, the maximum resources available per VM and resources allocated per VM details are displayed on the dialog. After selecting the DB Servers, click Save Changes.	

Setting	Description	Notes
Autonomous VM Cluster resources	VM count or Node Count: Denotes the number of database servers in the Exadata infrastructure. This is a read-only value. This value depends on the number of DB Servers selected for the AVMC. ECPU count per VM or node: Specify the ECPU count for each individual VM. The minimum value is 40 ECPUs per VM. Maximum number of Autonomous Container Databases: The number of ACDs specified represents the upper limit on ACDs. These ACDs must be created separately as needed. ACD creation also requires 8 available ECPUs or 2 available OCPUs per node. Database memory per CPU (GB): The memory per CPU allocated for the Autonomous AI Databases in the Autonomous VM Cluster. Allocate Storage for Local Backups: For Exadata	VM and node are used interchangeably between Oracle Exadata Cloud@Customer and Oracle Public Cloud deployments. As you configure the resource parameters, the aggregate resources needed to create the Autonomous Exadata VM Cluster and the values based on your selection are displayed below the resource configuration section. See Formula for sizing AVMC resources for more details.

Setting	Description	Notes
	Cloud@Custom er, you can check this option to configure the Exadata storage to enable local database backups. Database storage (TB): Data storage allocated for Autonomous AI Database creation in the Autonomous VM Cluster. The minimum value is 5TB.	

Setting	Description	Notes
Network Settings	Virtual cloud network: The virtual cloud network (VCN) in which you want to create the new Autonomous Exadata VM Cluster. Subnet: A subnet within the above selected VCN for the new Autonomous Exadata VM Cluster. Optionally, you can use network security groups to control traffic. To do so, Select the corresponding check-box and choose a Network Security Group from the select list. Click + Another Network Security Group to add more network security groups. In Oracle Public Cloud deployments, you can use IPv4/IPv6 dual-stack networking, enabling both IPv4 and IPv6 addresses. You can enable this option while creating a VCN, and while creating a subnet, and you can use the corresponding subnet (which has both IP4 and IP6 addresses)	APPLIES TO:  Oracle Public Cloud only

Setting	Description	Notes
	while provisioning an AVMC.	
VM cluster network	The VM cluster network in which to create the new Autonomous Exadata VM Cluster.	APPLIES TO:  Exadata Cloud@Customer only
Automatic maintenance	Optionally, you can configure the automatic maintenance schedule. On Oracle Public Cloud, Click Modify maintenance. On Exadata Cloud@Customer, Click Modify maintenance schedule. You can then change the maintenance schedule from the default (No preference, which permits Oracle to schedule maintenance as needed) by selecting Specify a schedule and then selecting the months, weeks, days and hours for the schedule. You can also set a lead time to receive a notification message about an upcoming maintenance from Oracle. When you are finished, save your changes.	For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable.

Setting	Description	Notes
License Type	The license type you wish to use for the new Autonomous Exadata VM Cluster. You have the following options: Bring your own license: If you choose this option, make sure you have proper entitlements to use for new service instances that you create. License included: With this choice, the cost of the cloud service includes a license for the Database service.	Your choice for the license type affects metering for billing.
Show/hide advanced options	By default, advanced options are hidden. Click Show advanced options to display them.	-

Setting	Description	Notes
Advanced options: Management	If you want to set a time zone other than UTC or the browser-detected time zone, then select the Select another time zone option, select a Region or country , and select the corresponding Time zone. If you do not see the region or country you want, select Miscellaneous and choose an appropriate Time zone.	The default time zone for the Autonomous Exadata VM Cluster is UTC, but you can specify a different time zone. The time zone options are those supported in both the <code>Java.util.TimeZone</code> class and the Oracle Linux operating system. You can not modify the time zone settings for an Autonomous Exadata VM Cluster already provisioned. If needed, you can create a service request in My Oracle Support. For help with filing a support request, see Create a Service Request in My Oracle Support .
Advanced options: Listener	If you want to choose a Single Client Access Name (SCAN) listener port for Transport Layer Security (TLS) and non-TLS from a range of available ports, enter a port number for Non-TLS port or TLS port in the permissible range (1024 - 8999). You can also choose mutual TLS (mTLS) authentication by selecting the Enable mutual TLS (mTLS) authentication check-box.	The default port number is 2484 for TLS and 1521 for non-TLS authentication modes. The port numbers 1522, 1525, 5000, 6100, 6200, 7060, 7070, 7879, 8080, 8181, 8888, and 8895 are reserved for special purposes and <i>can not be used</i> as Non-TLS or TLS port numbers. The SCAN listener ports can not be changed after provisioning the AVMC resource. As ORDS certificates are one-way TLS certificates, choosing between one-way TLS and mutual TLS (mTLS) is applicable only to database TLS certificates .

Setting	Description	Notes
Advanced options: Security attributes	Add a security attribute to control access for your AVMC resource using Zero Trust Packet Routing (ZPR) policies. To add a security attribute, select a Namespace , and provide the Key and Value for the security attribute. Click Add security attribute to add additional security attributes.	APPLIES TO:  Oracle Public Cloud only To select a security attribute at this step, you must already have set up security attributes with Zero Trust Packet Routing. For more information, see About Zero Trust Packet Routing . You also can add security attributes for existing AVMCs also. See Configure Zero Trust Packet Routing (ZPR) for an AVMC for details.
Advanced options: Tags	If you want to use tags, add tags by selecting a Tag Namespace , Tag Key , and Tag Value .	Tagging is a metadata system that allows you to organize and track resources within your tenancy. See Tagging Dedicated Autonomous AI Database Cloud Resources .

6. Optionally, you can save the resource configuration as a stack by clicking **Save as Stack**. You can then use the stack to create the resource through the Resource Manager service.

Enter the following details on the **Save as Stack** dialog, and click **Save**.

- **Name:** Optionally, enter a name for the stack.
- **Description:** Optionally, enter a description for this stack.
- **Save in compartment:** Select a compartment where this Stack will reside.
- **Tag namespace, Tag key, and Tag value:** Optionally, apply tags to the stack.

For requirements and recommendations for Terraform configurations used with Resource Manager, see [Terraform Configurations for Resource Manager](#). To provision the resources defined in your stack, [apply the configuration](#).

7. Submit your details to create the Autonomous Exadata VM Cluster.

Formula for sizing AVMC resources

The details of the formula used for sizing the Autonomous VM Cluster resources is given below.

The total resources requested for Autonomous Exadata VM Cluster are calculated with a simple formula in the console as shown below:

[Values based on user's input + Values based on internal calculation and overheads] x VM count

Table 6-1 The details of the calculation for each resource is as follows:

Name	Formula shown in the OCI Console	Example:
ECPU count	ECPU count per VM x VM count	ECPU Count 80 = ECPU count per VM (40) x VM count (2)
Memory	[Total ECPU count x Memory per ECPU] + Internal database service memory	Memory 246 GB = [Total ECPU count (80) x Memory per ECPU (2)] + Internal database service memory (86)
Local Storage	Local storage for one ACD + Internal database service local storage	Local storage 682 GB = Local storage for one ACD (100) + Internal database service local storage (582)
Exadata storage	Database Storage + Internal database service Exadata storage	Database Storage 6.55 TB = Database Storage (5) + Internal database service Exadata storage (1.55)

A more detailed explanation of the exact formula used which includes the internal sizing is given below.

- Total ECPU in AVM Cluster = ECPU count per VM x VM count
- Total Memory in AVM Cluster = [Database Memory per VM: (ECPU count per VM x Database memory per ECPU) + Internal service memory: 40GB] x 1.02 x VM Count
- Total Local Storage in AVM Cluster = (Autonomous VM Cluster storage + (u02 filesystem size x 1.03) + 2 GB) x VM Count
 - Where Autonomous VM Cluster storage value depends on the Exadata system model used, as follows:
 - * X8M, X9M, X10M, X11M systems: 184GB
 - * X7-2 and X8-2 systems: 137GB
 - And u02 filesystem size = 100GB + (50GB x Max Number of ACDs)
- Total Exadata Storage in AVM Cluster = [Database Storage (TB) + internal database storage (GB) + 150 GB] x data multiplier where:
 - internal database storage = 50 GB x Max number of ACDs x VM Count
 - data multiplier is 1.25 for Oracle Public Cloud deployments. For Exadata Cloud@Customer deployments, the value of data multiplier is 1.25 if local backups are disabled and 2.5 if local backups are enabled.

Manage Autonomous Exadata VM Clusters

An Autonomous Exadata VM Cluster resource provides the link between the Exadata Infrastructure resource and the Autonomous Container Database resources in Oracle Autonomous AI Database on Dedicated Exadata Infrastructure.

The maximum number of Autonomous Exadata VM Clusters (AVMC) and Autonomous Container Databases (ACD) you can create on an given Exadata Infrastructure varies based on the generation of hardware, as described in Resource Limits.

View a List of Autonomous Exadata VM Clusters

You can view a list of all Autonomous Exadata VM Clusters in a Compartment or of just those in a particular Exadata Infrastructure resource.

Required IAM Policies

For Autonomous AI Database on Exadata Cloud@Customer:

```
inspect autonomous-vmclusters
```

For Autonomous AI Database on Oracle Cloud:

```
inspect cloud-autonomous-vmclusters
```

View a List of Autonomous Exadata VM Clusters in a Compartment

You can view a list of all Autonomous Exadata VM Clusters in a Compartment by going to the Autonomous Exadata VM Clusters page.

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.

For instructions, see [Access Autonomous AI Database in the Oracle Cloud Infrastructure Console](#).

2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous Exadata VM Clusters**.

The Autonomous Exadata VM Clusters page is displayed, showing the list of Autonomous Exadata VM Clusters in your current Compartment. Choose the **Compartment** where you want to see the list of Autonomous Exadata VM Clusters.

3. The following details are displayed for each Autonomous Exadata VM Cluster resource in the list:
 - State of the resource such as Updating, Provisioning, Available, etc.
 - Available and total CPUs and percentage of used CPUs represented as a colour-coded bar chart. The colours on this bar represent the following:
 - **Less than 70%:** Green
 - **Between 70 to 90%:** Yellow
 - **Greater than 90%:** Red
 - Reclaimable CPUs. See [Resource Terminology](#) to understand what available, total, and reclaimable CPUs represent.
 - Available and total storage in TB and percentage of used storage represented as a colour-coded bar chart. The colours on this bar represent the following:
 - **Less than 70%:** Green
 - **Between 70 to 90%:** Yellow
 - **Greater than 90%:** Red
 - Provisionable ACDs, that is, the number of Autonomous Container Databases that can be created in the Autonomous VM Cluster.
 - Memory per CPU in GBs.
 - Date and time of creation of the resource.

View a List of Autonomous Exadata VM Clusters in an Exadata Infrastructure Resource

You can view a list of the Autonomous Exadata VM Clusters in a particular Exadata Infrastructure resource from its **Details** page:

1. Go to the **Details** page of the Exadata Infrastructure resource whose Autonomous Exadata VM Clusters you want to view. See [View Details of an Exadata Infrastructure Resource](#).

The **Details** page for the chosen Exadata Infrastructure resource is displayed.

2. Click **Autonomous Exadata VM clusters** to view a list of Autonomous Exadata VM Clusters in the selected Compartment that use the chosen Exadata Infrastructure resource.
3. The following details are displayed for each Autonomous Exadata VM Cluster resource in the list:
 - State of the resource such as Updating, Provisioning, Available, etc.
 - Available and total CPUs and percentage of used CPUs represented as a colour-coded bar chart. The colours on this bar represent the following:
 - **Less than 70%:** Green
 - **Between 70 to 90%:** Yellow
 - **Greater than 90%:** Red
 - Reclaimable CPUs. See [Resource Terminology](#) to understand what available, total, and reclaimable CPUs represent.
 - Available and total storage in TB and percentage of used storage represented as a colour-coded bar chart. The colours on this bar represent the following:
 - **Less than 70%:** Green
 - **Between 70 to 90%:** Yellow
 - **Greater than 90%:** Red
 - Provisionable ACDs, that is, the number of Autonomous Container Databases that can be created in the Autonomous VM Cluster.
 - Memory per CPU in GBs.
 - Date and time of creation of the resource.

Features of the List of Autonomous Exadata VM Clusters

You can perform a variety of actions in the list of Autonomous Exadata VM Clusters.

Action	Steps
Create an Autonomous Exadata VM Cluster	Click Create Autonomous Exadata VM Cluster and follow the instructions in Create an Autonomous Exadata VM Cluster .
View details of a listed Autonomous Exadata VM Cluster	In the list of Autonomous Exadata VM Clusters, click the name of the one whose details you want to view.
Move a listed Autonomous Exadata VM Cluster to a different compartment	In the list of Autonomous Exadata VM Clusters, click the  menu in the row, where you want to move the resource. Click Move resource and choose the new compartment for the Autonomous Exadata VM Cluster resource.

Action	Steps
Copy the OCID of a listed Autonomous Exadata VM Cluster	In the list of Autonomous Exadata VM Clusters, click the  menu in the row of the one whose OCID you want to copy and then click Copy OCID . The OCID is copied to your clipboard and the menu item changes to Copied .
Add security attributes	APPLIES TO:  Oracle Public Cloud only In the list of Autonomous Exadata VM Clusters, click the  menu in the row of the one you want to add security attributes, click Add security attributes .
Add tags to a listed Autonomous Exadata VM Cluster	In the list of Autonomous Exadata VM Clusters, click the  menu in the row of the one you want to apply tags to and then click Manage tags .
Terminate a listed Autonomous Exadata VM Cluster	In the list of Autonomous Exadata VM Clusters, click the  menu in the row of the one you want to terminate and then click Terminate . Then, in the Terminate Autonomous Exadata VM Cluster dialog: • Read the warning about terminating an Autonomous Exadata VM Cluster. • Enter the name of the Autonomous Exadata VM Cluster. • Submit your values to start the termination process.

View Details of an Autonomous Exadata VM Cluster

Follow these steps to view detailed information about an autonomous Exadata VM cluster.

Required IAM Policies

For Autonomous AI Database on Exadata Cloud@Customer:

```
inspect autonomous-vmclusters
```

For Autonomous AI Database on Oracle Public Cloud:

```
inspect cloud-autonomous-vmclusters
```

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous Exadata VM Clusters**.
You can see a list of Autonomous Exadata VM Clusters in your current Compartment.
3. Choose the **Compartment** where you want to see the list of Autonomous Exadata VM Clusters.
The list of Autonomous Exadata VM Clusters refreshes to show those in the selected Compartment.

4. Click the name of the Autonomous Exadata VM Cluster whose details you want to view.

The **Details** page for the Autonomous Exadata VM Cluster is displayed.

In Multicloud deployments, you can view the Multicloud information section with the following details : Subscription type, Subscription, Multicloud Zone, and Identity connector. If you're in an AWS Multicloud deployment, you additionally have the option to enable AWS KMS.

Actions Allowed

You can perform a variety of actions on the Details page for an Autonomous Exadata VM Cluster.

Action	Steps
View details of the Autonomous Exadata VM Cluster's Exadata Infrastructure resource	Click the name of the Exadata Infrastructure resource in the Infrastructure section.
Copy the Autonomous Exadata VM Cluster's OCID	In the General information section, click Copy in the OCID field.
Edit the Autonomous Exadata VM Cluster's maintenance schedule	In the Maintenance section, click Edit in the Maintenance schedule or Maintenance details field.
View upcoming and past maintenance	Click Maintenance . A list of upcoming planned maintenance activities, if any, is displayed. To view past maintenance activities, click Maintenance history .
Terminate the Autonomous Exadata VM Cluster	Click Terminate under Actions . Follow the instructions in Terminate an Autonomous Exadata VM Cluster.
Add tags to the Autonomous Exadata VM Cluster	On Oracle Public Cloud, click Manage tags under Actions . On Exadata Cloud@Customer, click Add tags under Actions .

Note

You have another way to add Tags. Click **Tags** and click **Add** to add a new tag.

View the tags applied to the Autonomous Exadata VM Cluster	Click the Tags tab.

Action	Steps
Enable or Disable AWS KMS	In Autonomous AI Database on Oracle Database@AWS, you can enable AWS Key Management Service (AWS KMS) to manage Master Encryption Keys. Enable AWS KMS: In the Multicloud information section, Click Enable in the AWS KMS field. Note AWS KMS can not be enabled if the AVMC has existing ACDs running with other Key Management types. Disable AWS KMS: You can disable AWS KMS as long as AVMC is empty and does not have any ACD in it. In the Multicloud information section, Click Disable in the AWS KMS field.
Manage Certificates	Click Manage certificates and follow instructions outlined in Manage Security Certificates for an Autonomous Exadata VM Cluster Resource .
Move Autonomous Exadata VM Cluster to a different compartment	Under Actions , click Move resource . In the new dialog that opens, choose the new compartment and submit the changes.
Add security attributes	APPLIES TO:  Oracle Public Cloud only Click Security. Click Add security attribute. Specify namespace, key and value to control access for your resources. Note You can also add security attributes by clicking Add security attributes under Actions.
View resource allocation details	In the Resource allocation section, click View Details . For additional information, see Resource Usage Tracking.
Scale up or scale down the resources within the Autonomous Exadata VM Cluster	Under Actions , click Scale Autonomous VM Cluster . Follow the instructions in Manage Autonomous Exadata VM Cluster Resources.

Resources Listed

You can view details about the following resources associated with this AVMC from the AVMC details page:

- **Autonomous Container Databases:** All the Autonomous Container Databases (ACD) in the selected compartment that use this AVMC are listed with details such as their state, role in an Autonomous Data Guard setup (Primary or standby), number of reclaimable CPUs, and creation date. You can also see the CPU value of the largest Autonomous AI Database that can be provisioned in this container database. To know more about these details, see Resource Terminology.
- **Autonomous Virtual Machines:** All the Autonomous Virtual Machines (DB Servers) used to deploy this AVMC are listed with details such as their name, state, CPUs, memory (GB), local storage (GB), client IP address, and DB server name.
- **Work Requests:** All the work requests associated with this AVMC are listed along with their state and other details.

You can also perform various actions on the ACDs listed. They are:

Actions	Steps
Create an Autonomous Container Database	Click Create Autonomous Container Database and follow the instructions listed in Create an Autonomous Container Database.
View details of a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the name of the one whose details you want to view.
Copy the OCID of a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the  menu in the row of the one whose OCID you want to copy and then click Copy OCID . The OCID is copied to your clipboard and the menu item changes to Copied.
Add tags to a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the  menu in the row of the one you want to apply tags to and then click Manage tags .
Move a listed Autonomous Container Database to a different compartment	In the list of Autonomous Container Databases, click the  menu in the row in the row where you want to move the resource and click Move resource . In the new dialog that opens, choose the new compartment and click Move resource .
Restart a listed Autonomous container Database resulting in a rolling restart of all Autonomous AI Databases in this container.	In the list of Autonomous Container Databases, click the  menu in the row where you want to restart the ACD and click Restart . On the Restart dialog, type the name of the Autonomous Container Database for confirmation, and click Restart .
Terminate a listed Autonomous Container Database	In the list of Autonomous Exadata VM Clusters, click the  menu in the row of the one you want to terminate and then click Terminate . Then, in the Terminate Autonomous Container Database dialog: • Read the warning about terminating an Autonomous Container Database. • Enter the name of the Autonomous Container Database. • Submit your details to begin the termination process.

Change the License Type on an Autonomous VM Cluster

Follow these steps to update the license type of an autonomous Exadata VM cluster:

Required IAM Policies

use autonomous-vmclusters

inspect exadata-infrastructures

Procedure

Note

To change the license type of an AVMC deployed on Oracle Public Cloud, submit a service request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for instructions.

To change the license type of an AVMC deployed on Exadata Cloud@Customer:

1. Open the navigation menu. Under **Oracle Database**, click **Oracle Exadata Database Service on Cloud@Customer**.
2. Click **Autonomous Exadata VM Clusters**.
3. In the list of Autonomous Exadata VM Clusters, click the display name of the Exadata VM cluster you wish to administer.
4. Under **Actions**, click **Update License Type**.
5. On the Update License Type dialog box, choose one of the following license types.
 - **Bring Your Own License (BYOL):** Select this option if your organization already owns Oracle Database software licenses that you want to use on the VM cluster.
 - **License Included:** Select this option to subscribe to Oracle Database software licenses as part of Exadata Cloud@Customer.

Updating the license type does not change the functionality or interrupt the operation of the VM cluster.

6. Click **Save**.

Move an Autonomous Exadata VM Cluster to Another Compartment

Follow these steps to move an Autonomous Exadata VM cluster from one compartment to another.

APPLIES TO:  Oracle Public Cloud APPLIES TO:  Exadata Cloud@Customer

Note

As soon as you move an Autonomous Exadata VM Cluster to a different compartment, the policies that govern the new compartment apply immediately and affect access to the Autonomous Exadata VM Cluster. Therefore, both your and other Oracle Cloud users' access to it may change, depending on the policies governing the user account's access to resources. For example, a user may lose the ability to create Autonomous Container Databases in it, given its new compartment.

Required IAM Policies**For Autonomous AI Database on Exadata Cloud@Customer:**

use `autonomous-vmclusters` in the Autonomous Exadata VM Cluster's current compartment and in the compartment you are moving it to.

For Autonomous AI Database on Oracle Public Cloud:

use `cloud-autonomous-vmclusters` in the Autonomous Exadata VM Cluster's current compartment and in the compartment you are moving it to.

Procedure

1. Go to the **Details** page of the Autonomous Exadata VM Cluster that you want to move. See [View Details of an Autonomous Exadata VM Cluster](#).
2. Under **Actions**, click **Move resource**.
3. Select the new compartment.
4. Click **Move resource**.

Enable or Disable AWS KMS

In Autonomous AI Database on Oracle Database@AWS, you can enable or disable AWS KMS on an Autonomous Exadata VM cluster.

Register AWS KMS Key

To enable AWS KMS for your Autonomous Exadata VM cluster, you must first register the AWS KMS key in OCI.

1. From the OCI console, select **Oracle AI Database**, and then select **Database Multicloud Integrations**.
2. From the left menu, select **AWS Integration**, and then select **AWS Keys**.
3. Select the **Register AWS keys** button, and then complete the following substeps.
 - a. From the dropdown list, select the **Compartment** where your Exadata VM Cluster resides.
 - b. Under the **AWS keys** section, select your identity connector from the dropdown list.

Note

Ensure that the role associated with the connector has the **DescribeKey** permission on the key. This permission is required to successfully perform discovery.

- c. The **Key ARN** field is optional.
 - d. Click the **Discover** button.
4. Once the key is discovered, select the **Register** button to register the key in OCI.

OCI IAM Policy Requirements

Allow any-user to read oracle-db-aws-keys in compartment id <your-compartment-OCID>
where all { request.principal.type = 'cloudautonomousvmcluster' }

This policy allows Oracle-managed Cloud VM Clusters to read the oracle-db-aws-keys resource in your compartment. It grants the necessary permissions for the VM cluster (the principal of type cloudautonomousvmcluster) to access the AWS key metadata required for integrating with external key management or performing cross-cloud operations. Without this policy, the VM cluster would not be able to retrieve the keys needed to complete the configuration.

Enable AWS KMS

Follow these steps to enable AWS KMS on an Autonomous Exadata VM cluster.

When you enable AWS key management for your database, only AWS keys that are authorized for use with AVMC and registered with OCI can be used.

Procedure:

1. Go to the Details page of the Autonomous Exadata VM Cluster where you want to enable AWS KMS. See [View Details of an Autonomous Exadata VM Cluster](#).
2. In the Multicloud information section, Click **Enable** in the AWS KMS field.
3. Confirm that you wish to enable AWS KMS in the confirmation dialog and submit.

Disable AWS KMS

Follow these steps to disable AWS KMS on an Autonomous Exadata VM cluster. You can disable AWS KMS as long as the AVMC is empty and does not have any ACD in it.

Procedure:

1. Go to the Details page of the Autonomous Exadata VM Cluster where you want to disable AWS KMS. See [View Details of an Autonomous Exadata VM Cluster](#).
2. In the Multicloud information section, Click **Disable** in the AWS KMS field.
3. Confirm that you wish to enable AWS KMS in the confirmation dialog and submit.

Manage Security Certificates for an Autonomous Exadata VM Cluster Resource

You can rotate the SSL security certificates for Oracle REST Data Services (ORDS) and Oracle Autonomous AI Database listener from the **Details** page of an Autonomous Exadata VM Cluster (AVMC).

Required IAM Policies

For Autonomous AI Database on Exadata Cloud@Customer:

manage autonomous-vmclusters

For Autonomous AI Database on Oracle Public Cloud:

manage cloud-autonomous-vmclusters

Procedure

1. Go to the **Details** page of the AVMC resource for which you want to manage security certificates.

For instructions, see View Details of an Autonomous Exadata VM Cluster Resource

2. Click **Manage Certificates**.
3. The **Manage Certificates** dialog launches. Make the following selections on this dialog:

Setting	Description	Notes
Manage certificate type	Choose the security certificate type that you want to manage. You have the following options: Database SSL certificate: Choose this option to manage SSL certificate for Database client connections. ORDS SSL certificate: Choose this option to manage ORDS SSL certificate for Application Express (APEX) applications.	Rotating a <i>Database SSL certificate</i> is a disruptive operation and may cause databases to become unavailable to new connections <i>for the duration of the certificate update</i>. Refer to Certificate Management for requirements about downloading and replacing the client wallets. Rotating an <i>ORDS SSL certificate</i> requires restarting ORDS, and all existing application connections would be lost.

Setting	Description	Notes
Certificate generation type	Choose the certificate generation type. This field defaults to System generated. Optionally, you can choose Bring your own certificate and provide the following details: Certificate: To bring your own certificate, you must select a certificate from the select list, changing compartments if necessary. See notes for more details about creating certificates from the Oracle Cloud Infrastructure (OCI) console. Specify CA certificate: Optionally, you can specify a CA certificate with the following options: • Certificate authority: If this option is selected, select a certificate authority from the select list, changing compartments if necessary. • CA bundle: If this option is selected, select a CA bundle from the select list, changing compartments if necessary.	To bring your own certificate, you must first create the certificate using the Oracle Cloud Infrastructure (OCI) Certificate Service. For information on creating certificates, see Creating a Certificate. These certificates must be signed and must be in the PEM format, that is, their file extension must be .pem, .cer, or .crt only. If you are selecting a CA bundle for an ORDS SSL certificate, the bundle must only contain certificates which are part of the certificate chain.

4. Click **Save**.
5. Click the confirmation button on the dialog that appears.

View Resource Usage for an Autonomous Exadata VM Cluster

Follow these steps to view the Autonomous Exadata VM Cluster's resource usage details.

Required IAM Policies

For Autonomous AI Database on Exadata Cloud@Customer

```
inspect autonomous-vmclusters
```

For Autonomous AI Database on Oracle Public Cloud

```
inspect cloud-autonomous-vmclusters
```

Procedure

To view the resource usage visualizations for an AVMC:

- Go to the **Details** page of the Autonomous Exadata VM Cluster that you want to monitor. See View Details of an Autonomous Exadata VM Cluster.
- In the **Resource Allocation** section, you can get a quick overview of the resources allocated to this AVMC. You can see the following details:
 - Available CPUs

- Total CPUs
- Available Autonomous AI Database Storage (TB)
- Total Autonomous AI Database Storage (TB)
- Database Memory per CPU (GB)
- Provisionable ACDs

✓ **Tip**

To understand the above resource attributes, refer to Resource Terminology.

- To get more details and a comprehensive picture of the resource allocation and their usage for this AVMC and its ACDs, click the **View Details** link. The **Resource allocation details** page opens with two tabs:

- **Autonomous Exadata VM Cluster:** This tab provides the following information:

- * **Total resources allocated:** This section lists the latest values for the following resources allocated to this AVMC:

- * CPUs
- * Exadata Storage in TB
- * Local Storage in GB
- * Memory in GB
- * Maximum number of ACDs
- * Autonomous AI Database Storage in TB
- * Autonomous AI Database Memory per CPU in GB.

For Exadata Cloud@Customer deployments, you can also see if the Storage for local backups is enabled or disabled.

- * **Resource usage visualizations:** This section has graphical and tabular representations of the AVMC's resource usage.

✓ **Tip**

You can choose to see this information either in the graphical or tabular view by selecting **Chart view** or **Table view** from the drop-down list at the top-right corner of this section.

For more information on the details presented in the chart and tabular formats, refer to Resource Usage Visualizations.

- **Autonomous Container Database:** This tab lists the following details for all the ACDs, in the selected AVMC, created in any compartment in the tenancy:
 - * ACD's display name.
 - * CPU value of the largest provisionable Autonomous AI Database.
 - * The number of CPUs provisioned to Autonomous AI Database.
 - * The number of reserved CPUs.
 - * The number of reclaimable CPUs.

Manage Autonomous Exadata VM Cluster Resources

You can add or remove resources to an already provisioned Autonomous Exadata VM Cluster (AVMC).

Required IAM Policies

For Autonomous AI Database on Exadata Cloud@Customer:

`manage autonomous-vmclusters`

For Autonomous AI Database on Oracle Public Cloud:

`manage cloud-autonomous-vmclusters`

Procedure

1. Go to the **Details** page of the AVMC resource you want to scale up or scale down. For instructions, see [View Details of an Autonomous Exadata VM Cluster](#).
2. Under **Actions**, click **Scale Autonomous VM Cluster**.
3. In the **Scale Autonomous VM Cluster** dialog, adjust the sliders to increase or decrease the following resources:
 - CPU count per VM
 - Maximum number of Autonomous Container Databases
 - Database storage (TB)

The minimum and maximum values on the sliders are the smallest and largest values the resources can scale to.

Note

Modifying CPU count per VM or the maximum number of Autonomous Container Databases of an AVMC triggers a rolling restart of the AVMC. This results in restarting all the ACDs and Autonomous AI Databases created in that AVMC.

4. Save your changes.
5. To confirm a rolling restart, enter the AVMC name in the **Confirm rolling restart** dialog, and click **Confirm**.

Note

In case of an ongoing maintenance activity on this AVMC or the ACDs and Autonomous AI Databases within it, your scale request fails with an appropriate message.

Terminate an Autonomous Exadata VM Cluster

Follow these steps to terminate an Autonomous Exadata VM cluster.

Required IAM Policies

For Autonomous AI Database on Exadata Cloud@Customer:

manage autonomous-vmclusters

For Autonomous AI Database on Oracle Public Cloud:

manage cloud-autonomous-vmclusters

Procedure

1. Go to the **Details** page of the Autonomous Exadata VM Cluster that you want to terminate. See View Details of an Autonomous Exadata VM Cluster.
2. Click **Terminate** under **Actions**.
3. Confirm that you wish to terminate your Autonomous Exadata VM Cluster in the confirmation dialog and submit.

Manage Exadata Infrastructure Maintenance

Oracle schedules and performs all patching and other maintenance operations on all the infrastructure resources of an Autonomous AI Database. At the same time, it also provides you with various options to customize, view, and reschedule maintenance events for the different infrastructure resources, including Exadata Infrastructure resources.

Change the Maintenance Schedule of an Autonomous Exadata VM Cluster

Note

You can not change the maintenance schedule of an AVMC created on the Exadata Infrastructure resources in Oracle Public Cloud before the launch of the multiple VM Autonomous AI Databases feature. Please create a service request in My Oracle Support if you need to address this limitation. For help with filing a support request, see Create a Service Request in My Oracle Support.

Required IAM Policies

use autonomous-vmclusters

inspect exadata-infrastructures

To change the maintenance schedule of an Autonomous Exadata VM Cluster:

1. Go to the **Details** page of the Autonomous Exadata VM Cluster whose maintenance schedule you want to change.
For instructions, see View Details of an Autonomous Exadata VM Cluster.
2. Click the **Edit** link in the Maintenance schedule field.
3. On the **Automatic Maintenance Schedule** card, change settings as desired.
 - Select **No Preference** to permit Oracle to schedule maintenance as needed
 - Select **Specify a Schedule** to restrict when Oracle can schedule maintenance. You can choose your preferred month, week, weekday, start time, and lead time for the Autonomous Exadata VM Cluster maintenance.

**Tip**

For guidance on choosing a custom schedule, see [Settings in Maintenance Schedule](#) that are Customizable.

4. When finished, save the changes.

View and Manage Scheduled Maintenance of an Autonomous Exadata VM Cluster

You view and manage scheduled maintenance of an Autonomous Exadata VM Cluster resource from its **Details** page.

Required IAM Policies

For Autonomous AI Database on Exadata Cloud@Customer

use `autonomous-vmclusters`

inspect `exadata-infrastructures`

For Autonomous AI Database on Oracle Public Cloud:

use `cloud-autonomous-vmclusters`

inspect `cloud-autonomous-vmclusters`

Procedure

1. Go to the **Details** page of the Autonomous Exadata VM Cluster whose maintenance schedule you want to change.

For instructions, see [View Details of an Autonomous Exadata VM Cluster](#).

2. Click **Maintenance**.

The **Maintenance** page lists any upcoming planned maintenance.

For more information about the details displayed in the **Maintenance** page, see [Scheduled Maintenance Details](#).

3. For each maintenance event listed in the **Maintenance** page, you can perform the management operations listed in [Management Operations on a Scheduled Maintenance](#), provided that the event is not already in progress.

When you reschedule an Autonomous Exadata VM Cluster maintenance event that is already scheduled, Oracle may place it in a queue if its Exadata Infrastructure resource or Autonomous Container Databases are:

- Already undergoing a maintenance update, or
- Scheduled for a maintenance activity simultaneously as the Autonomous Exadata VM Cluster.

For more information about automatic queuing of maintenance events, see [Automatic Queuing of Maintenance Events](#).

View Past Maintenance of an Autonomous Exadata VM Cluster

You view past maintenance of an Autonomous Exadata VM Cluster from its **Details** page.

Required IAM Policies

For Autonomous AI Database on Exadata Cloud@Customer

`inspect autonomous-vmclusters`

For Autonomous AI Database on Oracle Public Cloud

`inspect cloud-autonomous-vmclusters`

Procedure

1. Go to the **Details** page of the Autonomous Exadata VM Cluster whose maintenance schedule you want to change.
For instructions, see [View Details of an Autonomous Exadata VM Cluster](#).
2. Click **Maintenance History** to see a list of past maintenance.

Create and Manage Autonomous Container Databases

Create an Autonomous Container Database

You create an Autonomous Container Database from the **Autonomous Container Databases** page.

Required IAM Policies

Deployment Choice	IAM Policies
Oracle Public Cloud and Multicloud	<code>manage autonomous-container-databases</code> <code>use cloud-exadata-infrastructures</code> <code>use cloud-autonomous-vmclusters</code>
Exadata Cloud@Customer	<code>manage autonomous-container-databases</code> <code>use exadata-infrastructures</code> <code>use autonomous-vmclusters</code> <code>use backup-destinations</code>

Minimum Resource Requirements

To create *one Autonomous Container Database*, you need **at least**:

- 8 ECPUs or 2 OCPUs per node
- 50GB local storage per node

Related Live Labs

For a "try it out" alternative that demonstrates these instructions, see *Lab 6: Provisioning an Autonomous Container Database* in *Oracle Autonomous AI Database Dedicated for Fleet Administrators Workshop*.

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of resource types, click **Autonomous Container Database**.
The list of Autonomous Container Databases in your current Compartment is displayed.
3. Choose the **Compartment** where you want to create an Autonomous Container Database.
The list of Autonomous Container Databases refreshes to show those in the selected Compartment.
4. Click **Create Autonomous Container Database**.
The Create Autonomous Container Database page is displayed.
5. Fill out the Create Autonomous Container Database page with the following information:

Setting	Description	Notes
Compartment	Select a compartment to host the Autonomous Exadata VM Cluster.	
Display name	Enter a user-friendly description or other information that helps you easily identify the resource.	The display name does not have to be unique and you can change it whenever you want. Avoid entering confidential information.
Container Database Name	Enter a name for the container database. It can contain only letters and numbers. Starting with a letter and a maximum of 30 characters.	The container database name must be unique across the Autonomous Exadata VM Cluster.
 Tip The container database name is used to name the Oracle Key Vault's (OKV) wallet. You can use this name to locate the OKV wallet associated with your Autonomous Container Database on your OKV console.		
Exadata Infrastructure	The Exadata Infrastructure to host the new Autonomous Container Database.	
Autonomous Exadata VM Cluster	The Autonomous Exadata VM Cluster to host the new Autonomous Container Database.	

Setting	Description	Notes
Container Database Software Version	Oracle Database software version for the Autonomous Container Database. Depending on your preference, select one of the following options: • Select version from base images: When this option is selected, choose an Oracle Database software version from the Select base image list. The available options are 19c or 26ai. For all new Autonomous Exadata VM Clusters provisioned after April 7, 2026, users will be able to provision both 19c and 26ai database software versions in the same cluster. • Custom database software image: With this option, you can choose a custom image from the Choose custom image dialog.	See Create an Autonomous AI Database Software Image to know how to create custom software images. While selecting version from base images, you can choose either the latest Oracle Database software version or its immediate predecessor. For example: Suppose the latest Oracle Database version supported by Autonomous AI Database is 19.26.0.1.0. Then, the Select base image drop-down lists 19.26.0.1.0 and 19.25.0.1.0. for you to choose.
Configure automatic maintenance	The panel to configure automatic maintenance displays the following default settings: • Maintenance method: Rolling with time-zone file update disabled. • Container database maintenance version: Next release update (RU) • Maintenance schedule: No schedule preference specified. Optionally, you can configure a maintenance preference or schedule by clicking Modify maintenance schedule that launches the Edit automatic maintenance dialog.	DST stands for Daylight Savings Time.

Setting	Description	Notes
Edit automatic maintenance	Configure maintenance method: Choose between Rolling or Non-rolling maintenance methods. Optionally, you can also select Enable time-zone update. See Service Maintenance Types for more information. Configure container database maintenance version: Choose either Next RU or Latest RU to configure the next maintenance version of the ACD. Configure automatic maintenance schedule: Optionally, you can change the maintenance schedule from the default (No preference, which permits Oracle to schedule maintenance as needed) to Custom schedule in Oracle Public Cloud and Specify a schedule in Exadata Cloud@Customer. For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable. Click Save to close this dialog.	Updating the time-zone file would require a complete downtime for the ACD and the associated Autonomous AI Databases. The downtime is dependent on the amount of data that is time-zone sensitive. All the RUs that include a time-zone update will be patched in the non-rolling maintenance method (with full system downtime) only. So, if you configure your maintenance to the rolling method and enable time-zone update, all the RUs that include a time-zone update are applied in the non-rolling method only. Only those RUs without a time-zone update are applied in the rolling method. In an Autonomous Data Guard configuration, the non-rolling maintenance method results in downtime for the primary and standby ACDs during their respective maintenance window until the patching completes.

Setting	Description	Notes
Configure backups: Enable automatic backups	By default, automatic backups are enabled for an ACD. Optionally, you can choose to disable them by deselecting the Enable automatic backups check box. While provisioning an ACD with Autonomous Data Guard, you can not disable the automatic backups. When you enable automatic backups, choose from the following options to determine how long backups are retained after terminating the ACD: - Retain backups as per the backup retention period - Retain backups for 72hours, then delete	If disabled for an ACD, automatic backups can be enabled anytime later from the Oracle Cloud Infrastructure (OCI) console by following the steps outlined in Edit Autonomous Container Database Backup Settings. However, once enabled you can not disable automatic backups for the ACD. If enabling automatic backups fail for some reason, the ACD provisioning also fails with an error message. As a workaround, you can provision the ACD with automatic backups disabled, and enable them from the ACD's Details page later. The setting you choose for retaining backups after ACD's termination (which is 72 hours or till the duration of the backup retention period of the ACD) does not apply to long-term backups.
Configure backups: Backup destination type	The ability to choose a backup destination while provisioning an ACD, and the supported backup destinations vary depending on the deployment platform. On Oracle Public Cloud deployments of the Autonomous AI Database, you can choose either Autonomous Recovery Service or Object Storage as the backup destination. The default is Object Storage and the recommended option is Autonomous Recovery Service. For Autonomous AI Database on Oracle Database@AWS, you can choose either Autonomous Recovery Service, OCI Object Storage or Amazon Simple Storage (AWS S3). The default is AWS S3 and the recommended option is Autonomous Recovery Service. See Backup destination for more details on the different backup destination types.	

Setting	Description	Notes
Configure backups: Backup retention period (in days)	After enabling automatic backups, specify a Backup retention period value to meet your needs. You can choose any value between 7 to 95 days. See Backup Retention Policy for more details on the backup retention policy values for different Autonomous AI Database deployments.	
Retention lock	If you choose to retain backups for the duration of the backup retention period, you can also enable retention lock. Enabling retention lock: - Provides a 14-day grace period to disable the retention lock, after which the retention lock setting becomes permanent. - Retains all the automatic database backups until the ACD backup retention period. - Retains the long-term backups until their retention period (which is set while creating a long-term backup). Allows you only to increase the retention period that is already set, guaranteeing the backups remain for the retention period. - Disallows you from deleting any backup before its retention period expires. - Deletes all the automatic backups upon reaching their retention period expiry. However, if you have configured NFS for storing the automatic backups and the underlying ACD is terminated, then you have to manually delete the backups. - Deletes the long-term backups after they expire. If you have configured NFS for storing the long-term backups and the ACD is terminated, then you have to manually delete the backups after they expire.	If you have chosen to retain backups only for 72 hours after the ACD is terminated, then the retention lock option is unavailable.

Setting	Description	Notes
Configure backups: Enable cross-region backup copy	Optionally, select Enable cross-region backup copy and specify a region for the backup copies.	APPLIES TO:  Oracle Public Cloud only Cross region backup copy cannot be enabled on Autonomous Data Guard enabled ACDs. Cross region backup copies are always stored in Object Storage irrespective of the backup destination (Object Storage or Autonomous Recovery Service) chosen for the ACD.
Contact Email	Optionally, you can add contact emails to receive operational notifications, announcements, and unplanned maintenance notifications regarding your Autonomous Container Database.	Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.
Show/hide advanced options	By default, advanced options are hidden. Click Show advanced options to display them.	
Advanced options: Management	Optionally, you can define a suitable value for the following resource management attributes to suit your needs: • Database split threshold (CPU): The CPU value beyond which an Autonomous AI Database will be opened across multiple nodes. The default value of this attribute is 64 for ECPUs and 16 for OCPUs. • Node failover reservation (%): Determines the percentage of CPUs reserved across nodes to support node failover. Allowed values are 0%, 25%, and 50%, with 50% being the default option. • Distribution affinity: Determines whether an Autonomous AI Database must be opened across a minimum or maximum of nodes. By default, Minimum nodes is selected with Maximum nodes being the other option. Optionally, select Enable shared server connections to support the Net services architecture.	If the Node failover reservation is set to 0%, your Autonomous AI Database may face a complete outage during VM failure and maintenance operations such as database patching and container database restart. The shared server architecture enables a database server to allow many client processes to share very few server processes, so the number of users that can be supported is increased. You cannot disable shared server architecture after provisioning the ACD. See Special-Purpose Connection Features for more details.

Setting	Description	Notes
Advanced options: Encryption Key	Optionally, you can configure the Autonomous Container Database to use customer-managed encryption keys instead of Oracle-managed encryption keys. Select Encrypt using a customer-managed key in this tenancy and one of the following options, depending on your deployment: • OCI Vault Service: With this option, choose a Vault and a Master Encryption Key from the lists. This option is only available on Oracle Public Cloud. • Oracle Key Vault: With this option, choose a Key Store from the list and enter an OKV Endpoint group name. • AWS KMS: In Autonomous AI Database deployed in AWS cloud regions, you can use AWS Key Management Service (AWS KMS) to manage Master Encryption Keys. If the AVMC is enabled with AWS KMS, then only the AWS KMS option is enabled under Customer managed keys and other options like OKV or OCI Vault Service is disabled. When you choose AWS KMS, the corresponding compartment and discovered keys are populated in the list. To discover newly added keys, you can click on Discover Key.	By default, Encrypt using Oracle-managed keys is selected. You need to create a key store to use encryption key with Oracle Key Vault. See Use Customer-Managed Keys in Oracle Key Vault for instructions. You can use customer-managed encryption keys with Autonomous Data Guard-enabled Autonomous Container Databases with the primary and standby databases located in different availability domains either within the same region or within two different regions. When entering the name of an OKV Endpoint group, the endpoint name must be in all uppercase and may include numbers, hyphens (-), and underscores (_) and start with an uppercase letter. For AWS KMS, Only the registered keys are listed. If your desired key is not visible, it may not have been registered yet. Click Register AWS KMS Key to discover and register it. You can select a key alias from the drop-down list. If no key alias is available, the drop-down list will display the key ID. For information about using customer-managed keys see About Master Encryption Keys.
Advanced options: Tags	If you want to use tags, add tags by selecting a Tag Namespace , Tag Key , and Tag Value .	You can not modify the time zone settings for an Autonomous Exadata VM Cluster already provisioned. If needed, you can create a service request in My Oracle Support. For help with filing a support request, see Create a Service Request in My Oracle Support.

6. Optionally, you can save the resource configuration as a stack by clicking **Save as Stack**. You can then use the stack to create the resource through the Resource Manager service.

Enter the following details on the **Save as Stack** dialog, and click **Save**.

- **Name:** Optionally, enter a name for the stack.

- **Description:** Optionally, enter a description for this stack.
- **Save in compartment:** Select a compartment where this Stack will reside.
- **Tag namespace, Tag key, and Tag value:** Optionally, apply tags to the stack.

For requirements and recommendations for Terraform configurations used with Resource Manager, see [Terraform Configurations for Resource Manager](#). To provision the resources defined in your stack, [apply the configuration](#).

7. Submit your details to create the Autonomous Container Database.

The list of Autonomous Container Databases refreshes to show the new Autonomous Container Database with a status of **Provisioning** until it is available.

Edit Autonomous Container Database Backup Settings

If automatic backups were disabled while provisioning an Autonomous Container Database (ACD), you can enable them later from the Oracle Cloud Infrastructure (OCI) console.

Note

Once enabled, automatic backups can not be disabled for an ACD.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the Autonomous Container Database whose backup settings you want to change.
For instructions, see [View Details of an Autonomous Container Database](#).
2. Under **Actions**, click **Edit backup settings** on Oracle Public Cloud and **Backup settings** on Exadata Cloud@Customer.
3. If automatic backups are disabled for this ACD, enable them by selecting the **Enable automatic backups** check-box, and choose appropriate values for the following settings:

Setting	Description	Notes
Backup destination type	Select a backup destination type and then specify options based on the selected type. On Oracle Public Cloud deployments of the Autonomous AI Database, you can choose either Autonomous Recovery Service or Object Storage as the backup destination. The default is Object Storage and the recommended option is Autonomous Recovery Service. For Autonomous AI Database on Oracle Database@AWS, you can choose either Autonomous Recovery Service, OCI Object Storage or Amazon Simple Storage (AWS S3). The default is AWS S3 and the recommended option is Autonomous Recovery Service. See Backup destination for more details on the supported backup destination types.	APPLIES TO: ● Exadata Cloud@Customer only At any given point in time, there can only be a maximum of two backup destinations associated with an ACD. See Change Backup Destinations for the allowed switches from one backup destination type to another.
Deletion option after Autonomous Container database termination	Specify the duration to retain backups after ACD is terminated.	When you enable automatic backups, choose from the following options to determine how long backups are retained after terminating the ACD: • Retain backups as per the backup retention period • Retain backups for 72hours, then delete The setting you choose for retaining backups after ACD's termination does not apply to long-term backups.

Setting	Description	Notes
Backup retention period (in days)	Specify a Backup retention period value to meet your needs. You can choose any value between 7 to 95 days.	On Oracle Public Cloud deployments, the backup retention policy value defaults to 15 days. On Exadata Cloud@Customer deployments, the backup retention policy value is as follows: • If the backup destination type is local, then the backup retention period is 7 days. • If the backup destination type is NFS or Object Storage, then the backup retention period is between 7 days and 95 days as set by the customer. • If the backup destination type is a Recovery Appliance, then the backup retention period is controlled by a policy set by the customer on the appliance. All the backups are automatically deleted after the backup retention period.

Setting	Description	Notes
Enable or Disable retention lock	If you choose to retain backups for the duration of the backup retention period, you can also enable retention lock.	Enabling retention lock: - Provides a 14-day grace period to disable the retention lock, after which the retention lock setting becomes permanent. - Retains all the automatic database backups until the ACD backup retention period. - Retains the long-term backups until their retention period (which is set while creating a long-term backup). Allows you only to increase the retention period that is already set, guaranteeing the backups remain for the retention period. - Disallows you from deleting any backup before its retention period expires. - Deletes all the automatic backups upon reaching their retention period expiry. However, if you have configured NFS for storing the automatic backups and the underlying ACD is terminated, then you have to manually delete the backups. - Deletes the long-term backups after they expire. If you have configured NFS for storing the long-term backups and the ACD is terminated, then you have to manually delete the backups after they expire. If you have chosen to retain backups only for 72 hours after the ACD is terminated, then the retention lock option is unavailable.

Setting	Description	Notes
Enable cross-region backup copy	Select Enable cross-region backup copy and specify a region for the backup copies.	APPLIES TO:  Oracle Public Cloud only After creating a remote backup, you can see the Remote backup region from the ACD Details page. You can click this link to open the list of remote backups. When you change the region for cross-region backups, a new backup will be taken. The backup from the previously selected region will be retained for 7 days and deleted after. Whenever you disable cross-region backups on an ACD, the existing backup in the remote region will be retained for 7 days & deleted after. In Oracle Public Cloud deployments, Cross region backup copies are always stored in Object Storage irrespective of the backup destination (Object Storage or Autonomous Recovery Service) chosen for the ACD. In Multicloud deployments, this is only applicable if the backup destination is Object Storage.

4. When finished, save your changes.

Configure and Manage Autonomous Data Guard

The Autonomous Data Guard feature of Autonomous AI Database on Dedicated Exadata Infrastructure enables you to keep your critical production databases available to mission critical applications despite failures, disasters, human error, or data corruption. This kind of capability is often called *disaster recovery*.

Enable Autonomous Data Guard on an Autonomous Container Database

You can enable Autonomous Data Guard from the **Details** page of an Autonomous Container Database.

Note

You cannot enable Autonomous Data Guard on an ACD with an active maintenance run scheduled within the next three days.

Required IAM Permissions

```
inspect cloud-autonomous-vmclusters
use autonomous-container-databases
```

Procedure**Note**

When an add standby ACD operation is in progress, any scheduled maintenance on that ACD will not begin until the add standby operation is complete.

1. Go to the **Details** page of the Autonomous Container Database for which you want to enable Autonomous Data Guard.

For instructions, see View Details of an Autonomous Container Database.

2. Click **Enable** under **Autonomous Data Guard** in **Autonomous Container Database information**.
3. Alternatively, you can also click **Add Standby** on **Autonomous Data Guard groups**.
4. Fill out the **Add Standby** dialog with the following information:

Setting	Description	Notes
Peer Autonomous Container Database compartment	Select the standby Autonomous Container Database compartment.	
Peer Autonomous Container Database name	Enter a name for the standby ACD.	You can add a standby ACD from AWS region to the already provisioned ACD in the OCI region. Alternatively, you can add a standby ACD in the OCI region to an already provisioned ACD in the AWS region.
Peer region	Select a region for the standby ACD.	The primary and secondary ACDs can also be deployed in different regions (cross-region).
Peer Exadata Infrastructure	Select the underlying Exadata Infrastructure resource for the standby ACD.	
Peer Autonomous Exadata VM Cluster (AVMC)	Select the parent AVMC for the standby ACD.	In Exadata Cloud@Customer deployments, if the parent AVMC has the 3rd NIC enabled, all Data Guard operations will travel only over this 3rd NIC.
Protection Mode	Select Maximum performance or Maximum availability from the drop-down list.	Maximum Performance is selected by default. For information about Autonomous Data Guard and guidance in choosing where to place the standby autonomous container database and which protection mode to use, see About Autonomous Data Guard and Autonomous Data Guard Configuration Options.

Setting	Description	Notes
Peer database backup configuration	Select a Backup Destination type from the drop-down list. On Oracle Public Cloud deployments of the Autonomous AI Database, you can choose either Autonomous Recovery Service or Object Storage as the backup destination. The default is Object Storage and the recommended option is Autonomous Recovery Service. For Autonomous AI Database on Oracle Database@AWS, you can choose either Autonomous Recovery Service, OCI Object Storage or Amazon Simple Storage (AWS S3). The default is AWS S3 and the recommended option is Autonomous Recovery Service.	
Peer database maintenance preference	Select the number of days for which the standby ACD maintenance will be scheduled before primary ACD maintenance because standby ACD is always patched before primary ACD.	This option is available only when the primary ACD has defined a custom maintenance schedule.

- Confirm to add the standby database.

Note

Once enabled, Autonomous Data Guard can only be disabled by terminating the standby ACD.

View the Status of an Autonomous Data Guard Configuration

You view the status of an Autonomous Data Guard configuration from the **Details** page of the primary or standby Autonomous Container Database in the configuration.

Required IAM Policies

inspect autonomous-container-databases

Procedure

- Go to the **Details** page of the primary or standby Autonomous Container Database in the Autonomous Data Guard configuration.

For instructions, see View Details of an Autonomous Container Database.

You can view the Autonomous Data Guard details such as its status, peer role, peer state, protection mode, and automatic failover setting under **Autonomous Data Guard** in the details page of **Autonomous AI Database**.

2. You can also view the Autonomous Data Guard details by clicking **Autonomous Data Guard Groups** in the details page of the corresponding **Autonomous Container Database**.
The **Autonomous Data Guard Groups** table displays information about the peer container database, the current apply lag and transport lag, state, and last role change and creation dates.

Add a Second Standby Autonomous Container Database

In an Autonomous Data Guard setup, you can add a second standby Autonomous Container Database (ACD) to the primary ACD. The second standby ACD must be in the same tenancy as the primary ACD.

Prerequisites

To be able to add a second standby ACD, the first standby ACD should not have automatic failover enabled. You must disable automatic failover on the first standby before adding the second standby and can re-enable later.

Required IAM Permissions

use autonomous-container-databases

Procedure

Note

- When an add standby ACD operation is in progress, any scheduled maintenance on that ACD will not begin until the add standby operation is complete.
- Adding a standby database requires an automatic non-rolling restart for the first standby database. The primary database is not affected by this non-rolling restart.

1. Go to the **Details** page of the Autonomous Container Database for which you want to add a second standby database.

For instructions, see View Details of an Autonomous Container Database.

2. Click **Add standby** in **Autonomous Data Guard groups**.
3. Fill out **Add standby** with the following information:

Setting	Description
Peer Autonomous Container Database compartment	Select the standby Autonomous Container Database compartment.

Setting	Description
Peer Autonomous Container Database name	Enter a name for the standby ACD.
 Note You can add a standby ACD from AWS region to the already provisioned ACD in the OCI region. Alternatively, you can add a standby ACD in the OCI region to an already provisioned ACD in the AWS region.	
Peer region	Select a region for the standby ACD.
Peer Exadata Infrastructure	Select the underlying Exadata Infrastructure resource for the standby ACD.
Peer Autonomous Exadata VM Cluster (AVMC)	Select the parent AVMC for the standby ACD.
Peer Database Backup Configuration	Select the backup destination type for the second standby database from the drop-down list. On Oracle Public Cloud deployments of the Autonomous AI Database, you can choose either Autonomous Recovery Service or Object Storage as the backup destination. The default is Object Storage and the recommended option is Autonomous Recovery Service. For Autonomous AI Database on Oracle Database@AWS, you can choose either Autonomous Recovery Service, OCI Object Storage or Amazon Simple Storage (AWS S3). The default is AWS S3 and the recommended option is Autonomous Recovery Service.

 Note

You cannot explicitly set maintenance preferences for the second standby ACD as it inherits these preferences from the first standby ACD of the primary ACD.

4. Click **Add standby**.

Switch Roles in an Autonomous Data Guard Configuration

You switch the roles of the primary and standby Autonomous Container Databases in an Autonomous Data Guard configuration from the **Details** page of the primary or standby Autonomous Container Database.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the *standby* ACD that you want to switch roles with the primary ACD in the Autonomous Data Guard configuration.

For instructions, see View Details of an Autonomous Container Database.

Note

You can not switch roles of the primary and standby ACDs in an Autonomous Data Guard configuration where the standby is in the **snapshot standby** role.

2. Under **Actions**, click **Switchover**.
3. Enter the ACD name in the confirmation dialog, and click **Switchover**.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure sets the state of the standby and its primary container databases to **Role Change in Progress** and begins the switchover operation, which causes the primary container database to assume the standby role and the standby container database to assume the primary role. Upon completion, the state of both container databases returns to **Active**.

Fail Over to the Standby in an Autonomous Data Guard Configuration

You fail over to the standby Autonomous Container Databases in an Autonomous Data Guard configuration from the **Details** page of the standby Autonomous Container Database.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the *standby* ACD to which you want to fail over in the Autonomous Data Guard configuration.

For instructions, see View Details of an Autonomous Container Database.

2. Under **Actions**, click **Failover**.
3. In case of a snapshot standby Autonomous Container Database, you see a message alerting you that the snapshot standby will be converted to physical standby after discarding all its local updates and applying data from your primary database. Click **Failover** to proceed.
4. Enter the ACD name in the confirmation dialog, and click **Fail over**.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure sets the state of the Standby container database to **Role Change in Progress** and begins the failover operation. Upon completion, the role of the Standby container database becomes **Primary** and the role of the Primary container database becomes **Disabled Standby** with the **Unavailable** state.

Reinstate the Disabled Standby in an Autonomous Data Guard Configuration

After a failover has occurred and the failed primary Autonomous Container Database assumes a disabled, standby role, you can reinstate the failed database to an enabled, standby role from its **Details** page.

In an Autonomous Data Guard setup with multiple standby databases and automatic failover:

- Manual failovers require you to manually reinstate the original primary database, which becomes the new standby database.
- Whenever an automatic failover occurs, Autonomous AI Database on Dedicated Exadata Infrastructure tries to reinstate the old primary as a standby. However, if that attempt fails, then it must be manually reinstated.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the *Disabled standby* ACD that you want to reinstate.

For instructions, see View Details of an Autonomous Container Database.

✓ Tip

The primary database that you failed over is labeled as "Disabled Standby" in the list of Autonomous Container Databases for a compartment.

2. Under **Actions**, click **Reinstate**.
3. Provide a confirmation to proceed with the reinstatement of the disabled standby ACD.

The states of the peer databases become **Role Change in Progress** until the reinstate action is complete. Upon completion, the role of the Disabled Standby container database becomes **Standby** and its state changes to **Available**.

Update Autonomous Data Guard Settings

You can update the settings of an Autonomous Data Guard from the **Details** page of the primary Autonomous Container Database in the configuration.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the primary Autonomous Container Database in the Autonomous Data Guard configuration.

For instructions, see View Details of an Autonomous Container Database.

2. Click **Update Autonomous Data Guard** under **Actions**.

The **Update Autonomous Data Guard** dialog displays the current settings for **Protection Mode** and **Automatic Failover**.

3. You can make the following updates from this dialog:
 - a. **Protection mode:** Select **Maximum performance** or **Maximum availability** from the drop-down list.
 - b. **Automatic failover:** If automatic failover is not enabled already, you can enable it by selecting **Enable automatic failover**. Similarly, you can deselect **Enable automatic failover** to disable automatic failover for this Autonomous Data Guard setup. If one of your standby databases is in the same region as the primary database and the second is in a different region, the local standby database will be prioritized over the remote standby as the automatic failover target. When you enable automatic failover, any of the standby databases will be considered for the automatic failover target.

Note

You can not enable Automatic Failover for databases with cross-region Autonomous Data Guard setup on Exadata Cloud@Customer deployments.

- c. **Fast start failover lag limit:** If automatic failover is enabled and the protection mode is Maximum Performance, the Fast start failover lag limit value is displayed in seconds. By default, this value is set to *30 seconds*, but you can change it to *any value between 5 and 3600 seconds*.
4. Save your changes.

On the Oracle Cloud Infrastructure console, the Autonomous Container Database state shows **UPDATING** until the updated Autonomous Data Guard settings are applied.

Convert Physical Standby to Snapshot Standby

You can convert a standby Autonomous Container Database to a snapshot standby in an Autonomous Data Guard setup from the **Details** page of the standby Autonomous Container Database in the configuration.

Required IAM Policies

use `autonomous-container-databases`

Procedure

1. Go to the **Details** page of the standby Autonomous Container Database in the Autonomous Data Guard configuration.

For instructions, see [View Details of an Autonomous Container Database](#).

2. Click **Convert to snapshot standby** under **Actions**.

Note

Converting to snapshot standby is not supported when automatic failover is enabled. You must disable automatic failover before converting to a snapshot standby. See [Update Autonomous Data Guard Settings](#) for instructions to disable automatic failover in an Autonomous Data Guard setup.

3. The **Convert to snapshot standby** dialog displays with options to use new Database services or primary Database services for the snapshot standby database connections.
 - **Use new Database services:** Click this option to connect to snapshot standby using new services that are active only in the snapshot standby mode.
 - **Use primary Database services:** Click this option if you wish to connect to snapshot standby database using the same services as the primary database.

Note

Activating primary database services on the snapshot standby database may result in snapshot standby connection requests forwarded to the primary database or vice-versa if you use incorrect database connect strings. Hence, you must be careful to use appropriate connect string while connecting to your primary and snapshot standby database when you choose to use primary Database services.

4. Click **Convert.**

On the Oracle Cloud Infrastructure console the Autonomous Container Database state shows **UPDATING** until the standby changes to snapshot standby.

Convert Snapshot Standby to Physical Standby

You can convert a snapshot standby Autonomous Container Database to a physical standby in an Autonomous Data Guard setup from the **Details** page of the standby Autonomous Container Database in the configuration.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the standby Autonomous Container Database in the Autonomous Data Guard configuration.

For instructions, see View Details of an Autonomous Container Database.

2. Click **Convert to physical standby** under **Actions**.
3. The **Convert to physical standby** dialog displays a message alerting you that converting the snapshot standby to physical standby will discard all its local updates and apply data from your primary database.
4. Click **Convert**.

On the Oracle Cloud Infrastructure console the Autonomous Container Database state shows **UPDATING** until the standby changes to physical standby.

Add a Cross Tenancy Standby Database

You can add an Autonomous Data Guard standby database that resides in a tenancy different from the primary database.

APPLIES TO:  Oracle Public Cloud only

Required IAM Policies

To create a cross tenancy standby database, you must ensure to meet the following requirements:

- Run the CLI or API commands to add the cross tenancy standby database in the destination tenancy.
- Define OCI Identity and Access Management groups and policies on the source and destination tenancies so that you can run commands to add the cross tenancy standby database in the destination tenancy and allow the destination tenancy to contact the

source tenancy where the primary database resides. When these policies are revoked, adding a cross tenancy standby database will not be allowed.

- On the **destination tenancy**, create a group (for example: *DestinationGroup*), and add the user(s) who will be allowed to add a cross tenancy standby database to this group. See [Using the Console to Create a Group](#) for guidance.
- On the **source tenancy**, create IAM policies to allow the group created in the destination tenancy (*DestinationGroup*) to add a cross tenancy standby database using the primary database from the source tenancy. See [Using the Console to Create a Policy](#) for guidance.

For example, you can define a policy to allow a user in the *DestinationGroup* of the *DestinationTenancy* read from a specific Autonomous AI Database instance in the specified compartment on the source tenancy as shown below:

```
define tenancy DestinationTenancy as ocid1.tenancy.oc1..unique_ID
define group DestinationGroup as ocid1.group.region1..unique_ID
admit group DestinationGroup of tenancy DestinationTenancy to read
autonomous-database-family in
    tenancy
```

Note

The policy only needs to allow read access on the source Autonomous AI Database instance to create a cross tenancy standby.

The above policy specifies the following:

- * Line 1: OCID of the destination tenancy where you are going to add the standby database.
 - * Line 2: OCID of the destination group to which the user who will create cross tenancy standby database belongs.
 - * Line 3: Specifies the destination group that can be allowed to read Autonomous AI Databases in the source tenancy.
- On the **destination tenancy**, create IAM policies to endorse a group to manage the primary database source on the source tenancy. See [Using the Console to Create a Policy](#) for guidance.

For example:

```
define tenancy SourceTenancy as ocid1.tenancy.oc1..unique_ID
endorse group DestinationGroup to manage autonomous-database-family in
tenancy SourceTenancy
```

The above policy specifies the following:

- * Line 1: OCID of the source tenancy OCID where the primary database resides.
- * Line 2: Specifies the destination group that can be allowed to manage Autonomous AI Databases in the source tenancy.

This policy discussed in the above example allows *DestinationGroup* to create Autonomous AI Databases and cross tenancy standby databases in the source

tenancy. See [IAM Permissions and API Operations for Autonomous AI Database](#) for more information and examples.

To add a cross tenancy standby database in the same region:

-
- [CLI](#)
 - [REST API](#)

CLI

You can create a standalone ACD in the source tenancy using CLI as shown below.

```
oci --debug db autonomous-container-database create
--cloud-autonomous-vm-cluster-id ocid1.cloudautonomousvmcluster.oc1.<region-
name>.unique_ID
--compartment-id ocid1.compartment.oc1.unique_ID --display-name clicrosdg
--patch-model RELEASE_UPDATES --service-level-agreement-type STANDARD
```

where:

- `--cloud-autonomous-vm-cluster-id`: The OCID of the target Autonomous Exadata VM Cluster where the new Autonomous Container Database (ACD) will be provisioned.

Alternatively, you can also use an existing standalone ACD and directly add the standby database as shown below.

On the tenancy where you want to add the standby database, that is, on the destination tenancy, use CLI to add a standby database as shown below:

```
oci --debug db autonomous-container-database add
--autonomous-container-database-id
ocid1.autonomoucontainerdatabase.oc1.<region-name>.unique_ID
--peer-autonomous-container-database-display-name clisecdg
--peer-cloud-autonomous-vm-cluster-id
ocid1.cloudautonomousvmcluster.oc1.<region-name>.unique_ID
--protection-mode MAXIMUM_PERFORMANCE
--peer-autonomous-container-database-compartment-id
"ocid1.compartment.oc1.unique_ID"
```

where:

- `--autonomous-container-database-id`: The OCID of the primary Autonomous Container Database to which you are adding a standby.
- `--peer-autonomous-container-database-display-name`: A user-friendly name for the new standby Autonomous Container Database in the destination tenancy.
- `--peer-autonomous-vm-cluster-id`: The OCID of the Cloud Autonomous VM Cluster in the destination tenancy where the standby database will be created.
- `--peer-autonomous-container-database-compartment-id`: The OCID of the compartment where the standby Autonomous Container Database will reside.
- `--protection-mode`: Defines the protection mode that will be used once the ACD is Data Guard enabled.

Once the command succeeds a work-request-id will be returned which can be used to track the progress of the standby database. See [autonomous-container-database](#) for more information.

Note

If you want the command to be run in a region which is not listed in the config file, then you need to set the required region in the command prompt before running the oci command.

For information about SDKs, see [Software Development Kits and Command Line Interface](#).

REST API

To add a cross tenancy standby database residing in the same region as the primary database using **REST API**, use `AutonomousContainerDatabases`.

The API call to create the standby is sent to the different tenancy in the same region.

You can create a standalone ACD in the source tenancy using REST API as shown below.

```
oci raw-request --http-method POST --target-uri https://database.us-ashburn-1.oraclecloud.com/20160918/autonomousContainerDatabases
--request-body '{
  "cloudAutonomousVmClusterId": "ocid1.cloudautonomousvmcluster.oc1.<region-name>.unique_ID",
  "compartmentId": "ocid1.compartment.oc1.unique_ID",
  "displayName": "cliapcrdg",
  "patchModel": "RELEASE_UPDATES",
  "serviceLevelAgreementType": "AUTONOMOUS_DATAGUARD",
}'
```

where:

- `cloudAutonomousVmClusterId`: The OCID of of the target Autonomous Exadata VM Cluster where the new Autonomous Container Database (ACD) will be provisioned.

Alternatively, you can also use an existing standalone ACD and directly add the standby database as shown below.

On the tenancy where you want to add the standby database, that is, on the destination tenancy, use REST API to add a standby database as shown below:

```
oci raw-request --http-method POST --target-uri
https://database.<region-name>.oraclecloud.com/../../../../
ocid1.autonomouscontainerdatabase.oc1.<region-name>../actions/addStandby
--request-body '{
  "peerCloudAutonomousVmClusterId":
"ocid1.cloudautonomousvmcluster.oc1.<region-name>.unique_ID",
  "peerAutonomousContainerDatabaseCompartmentId":
"ocid1.compartment.oc1.unique_ID",
  "peerAutonomousContainerDatabaseDisplayName": "cliapsec1",
  "protectionMode": "MAXIMUM_PERFORMANCE",
}'
```

where:

- `peerCloudAutonomousVmClusterId` : The OCID of the Cloud Autonomous VM Cluster in the destination tenancy where the standby database will be created.
- `peerAutonomousContainerDatabaseCompartmentId`: The OCID of the compartment where the standby Autonomous Container Database will reside.
- `peerAutonomousContainerDatabaseDisplayName`: A user-friendly name for the new standby Autonomous Container Database in the destination tenancy.
- `protectionMode`: Defines the protection mode that will be used once the ACD is Data Guard enabled.

See [AutonomousContainerDatabase](#) for additional information on the REST API.

For information about using the API and signing requests, see [REST APIs](#) and [Security Credentials](#).

To create a cross tenancy standby database in a remote region:

- [CLI](#)
- [REST API](#)

CLI

You can create a standalone ACD in the source tenancy (if you have not created one already) using CLI as shown below.

```
oci --debug db autonomous-container-database create
--cloud-autonomous-vm-cluster-id
ocid1.cloudautonomousvmcluster.oc1.<sourceregion-name>.unique_ID
--compartment-id ocid1.compartment.oc1.unique_ID --display-name clicrosdg
--patch-model RELEASE_UPDATES --service-level-agreement-type STANDARD
```

where:

- `--cloud-autonomous-vm-cluster-id`: The OCID of of the target Autonomous Exadata VM Cluster where the new Autonomous Container Database (ACD) will be provisioned.

Alternatively, you can also use an existing standalone ACD and directly add the standby database as shown below.

On the tenancy where you want to add the standby database, that is, on the destination tenancy in the remote region, use CLI to add a standby database as shown below:

```
oci --debug db autonomous-container-database add
--autonomous-container-database-id
"ocid1.autonomouscontainerdatabase.oc1.<sourceregion-name>.unique_ID"
--peer-autonomous-container-database-display-name "cliCrosstSec2"
--protection-mode "MAXIMUM_PERFORMANCE"
--peer-cloud-autonomous-vm-cluster-id
"ocid1.cloudautonomousvmcluster.oc1.<remoteregion-name>.uniqueID"
```

```
--peer-autonomous-container-database-compartment-id
"ocid1.compartment.oc1..uniqueID"
```

where:

- `--autonomous-container-database-id` : The OCID of the primary Autonomous Container Database in the source region to which you are adding a standby.
- `--peer-autonomous-container-database-display-name`: A user-friendly name for the new standby Autonomous Container Database in the destination region and in the destination tenancy.
- `--peer-autonomous-vm-cluster-id` : The OCID of the Cloud Autonomous VM Cluster in the destination region and in the destination tenancy where the standby database will be created.
- `-peer-autonomous-container-database-compartment-id`: The OCID of the compartment where the standby Autonomous Container Database will reside.
- `--protection-mode`: Defines the protection mode that will be used once the ACD is Data Guard enabled

Note

If the remote region is different from what is listed in the default config file, then you need to set the required remote region in the command prompt before running the `oci` command.

Once the command succeeds a `work-request-id` will be returned which can be used to track the progress of the standby database. See [autonomous-container-database](#) for more information.

For information about SDKs, see [Software Development Kits and Command Line Interface](#).

REST API

To add a cross tenancy standby database residing in a different region as the primary database using **REST API**, use `AutonomousContainerDatabases`.

You can create a standalone ACD in the source tenancy using REST API as shown below.

```
oci raw-request --http-method POST --target-uri https://database.us-
ashburn-1.oraclecloud.com/20160918/autonomousContainerDatabases --request-
body '{
  "cloudAutonomousVmClusterId":
"ocid1.cloudautonomousvmcluster.oc1.<sourceregion-name>.unique_ID",
  "compartmentId": "ocid1.compartment.oc1.unique_ID",
  "displayName": "cliapcrdg",
  "patchModel": "RELEASE_UPDATES",
  "serviceLevelAgreementType": "AUTONOMOUS_DATAGUARD",
}'
```

where:

- `cloudAutonomousVmClusterId`: The OCID of the target Autonomous Exadata VM Cluster where the new Autonomous Container Database (ACD) will be provisioned.

Alternatively, you can also use an existing standalone ACD and directly add the standby database as shown below.

On the tenancy where you want to add the standby database, that is, on the destination tenancy in the destination region, use REST API to add a standby database as shown below:

```
oci raw-request --http-method POST --target-uri
https://database.ap-chuncheon-1.oraclecloud.com/20160918/
autonomousContainerDatabases/
ocid1.autonomouscontainerdatabase.oc1.<source-region-name>../actions/
addStandby
--request-body '{
  "peerCloudAutonomousVmClusterId":
"ocid1.cloudautonomousvmcluster.oc1.<remote-region-name>.uniqueID",
  "peerAutonomousContainerDatabaseCompartmentId":
"ocid1.compartment.oc1.uniqueID",
  "peerAutonomousContainerDatabaseDisplayName": "cliapsec2",
  "protectionMode": "MAXIMUM_PERFORMANCE",
}'
```

where:

- `peerCloudAutonomousVmClusterId` : The OCID of the Cloud Autonomous VM Cluster in the destination tenancy where the standby database will be created.
- `peerAutonomousContainerDatabaseCompartmentId`: The OCID of the compartment where the standby Autonomous Container Database will reside.
- `peerAutonomousContainerDatabaseDisplayName`: A user-friendly name for the new standby Autonomous Container Database in the destination tenancy.
- `protectionMode`: Defines the protection mode that will be used once the ACD is Data Guard enabled.

See [AutonomousContainerDatabase](#) for additional information on the REST API.

For information about using the API and signing requests, see [REST APIs](#) and [Security Credentials](#).

Note

After submitting a request to add a cross tenancy standby database, the database Lifecycle State shows Updating. You cannot stop, start, restart, restore or move the Autonomous AI Database in this state.

Manage Customer Contacts for an Autonomous Container Database

You can add, edit, and remove customer contacts for an Autonomous Container Database from its **Details** page.

Required IAM Policies

use `autonomous-container-databases`

Procedure

1. Go to the **Details** page of the Autonomous Container Database for which you want manage customer contacts. For instructions, see View Details of an Autonomous Container Database.
2. On the **Details** page, Under **Actions**, click **Manage customer contacts**. The **Manage customer contacts** page opens, enabling you to perform the following actions:

Add Customer Contacts

You can provide up to ten contacts' email address for operational notifications and announcements. In Oracle Public Cloud, click **Add Contact** after entering each email address. In Exadata Cloud@Customer, click **+Another Contact** after entering details for one contact. Click **Save** to save your changes.

Note

Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.

Edit Customer Contacts

Select the email addresses that you are editing. Click **Save** after making the changes.

Delete Customer Contacts

Select the contact detail to be removed. Click **X** at the end of the row. The contact details are removed. Save your changes.

Move an Autonomous Container Database to a Different Compartment

You can move an Autonomous Container Database to a different Oracle Cloud Infrastructure compartment.

Note

As soon as you move an Autonomous Container Database to a different compartment, the policies that govern the new compartment apply immediately and affect access to the Autonomous Container Database. Therefore, both your and other Oracle Cloud users' access to it may change, depending on the policies governing the user account's access to resources. For example, a user may lose the ability to create Autonomous AI Databases in the Autonomous Container Database, given its new compartment.

Required IAM Policies

use `autonomous-container-databases` in the Autonomous Container Database's current compartment and in the compartment you are moving it to.

Procedure

1. Go to the **Details** page of the Autonomous Container Database you want to move.
For instructions, see View Details of an Autonomous Container Database.
2. Under **Actions**, click **Move resource**.
3. In the new dialog, select the new compartment.
4. Click **Move resource**.

Restart an Autonomous Container Database

You restart an Autonomous Container Database by clicking **Restart** on its **Details** page.

The restart of an Autonomous Container Database occurs in a rolling fashion, first stopping and starting one of the container database's database instances and then stopping and starting its other database instance. Thus, at no time are the container database or any of the Autonomous AI Databases created in it unavailable.

Note

You may experience performance degradation while restarting an Autonomous Container Database with Database In-Memory enabled Autonomous AI Databases. For additional information on Database In-Memory, see Database In-Memory.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the Autonomous Container Database you want to restart.
For instructions, see View Details of an Autonomous Container Database.
2. Click **Restart**.
3. In the **Restart** dialog:
 - a. Enter the name of the Autonomous Container Database.
 - b. Click **Restart** to begin the rolling restart operation.

Note

In an Autonomous Data Guard configuration, clicking **Restart** on a primary or standby database does not restart its peer ACDs.

Schedule a Quarterly Maintenance Update

You can schedule an on-demand maintenance to update RU (Release Update) along with the time-zone file or just the time-zone file for an Autonomous Container Database (ACD). You can also choose to update using an existing custom database software image.

Note

For on-demand maintenance updates with a time-zone file update, the standby ACD will be patched 3 days before the primary ACD in autonomous data guard enabled ACDs.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the Autonomous Container Database for which you want to schedule a maintenance update.

For instructions, see View Details of an Autonomous Container Database.

2. Under **Actions**, click **Schedule maintenance**.
3. Choose the following on the **Schedule maintenance** dialog that appears.

Setting	Description	Notes
Configure update type	Custom database software image: Click this tile to schedule an update using an existing custom database software image. You can select a Region, Compartment, and Database version to choose a Custom database software image. Release Update (RU): Click this tile to schedule a release update (RU) along with time-zone file update. This tile is disabled if your ACD is already patched to the latest RU. Time-zone file update: Click this tile to schedule time-zone file update only.	Time-zone files can only be updated in a non-rolling configuration method that involves full system downtime. If the selected Custom database software image has an updated Time-zone version, the time-zone of the ACD will be updated. Updating the time-zone requires mandatory downtime for the ACD and associated Autonomous AI Databases. The downtime duration depends on the amount of time-zone sensitive data. You cannot select a custom database software image of a previous RU. Selecting the Custom database software image for patching will cancel the current maintenance run if one is scheduled for your ACD.
Configure maintenance method	The Rolling method updates the ACD, one node at a time, with no downtime for the Autonomous AI Databases. The Non-rolling method shuts down and updates the ACD in parallel across all nodes. This method minimizes maintenance time but it requires full downtime for the ACD and all associated Autonomous AI Databases.	

Setting	Description	Notes
Enable time-zone update	Optionally, you can also select Enable time-zone update . Time-zone files can only be updated using the non-rolling configuration method.	If you configure your maintenance to the rolling method and enable time-zone update , all the RUs that include a time-zone update are applied in the non-rolling method only. Only those RUs without a time-zone update are applied in the rolling method. You cannot enable time-zone update if you are scheduling an update using a Custom database software image.
Schedule start time	Using the calendar tool, choose a date and time for the maintenance update.	

4. Submit the details to schedule the maintenance.

Note

The maintenance settings selected from the **Schedule maintenance** dialog apply to the maintenance update scheduled from this dialog only. These changes do not apply to the maintenance preferences defined at the ACD level.

Terminate an Autonomous Container Database

You terminate an Autonomous Container Database by clicking **Terminate** on its **Details** page.

Required IAM Policies

`manage autonomous-container-databases`

`use autonomous-exadata-infrastructures` if creating the Autonomous Container Database on Oracle Cloud

`use autonomous-vmclusters` if creating the Autonomous Container Database on Exadata Cloud@Customer

Procedure

1. Go to the **Details** page of the Autonomous Container Database you want to terminate.
For instructions, see View Details of an Autonomous Container Database.
2. Click **Terminate** under **Actions**.
3. In the **Terminate Autonomous Container** dialog:
 - a. Read the warning about terminating an Autonomous Container Database.
 - b. Enter the name of the Autonomous Container Database.
 - c. Confirm to begin the termination process.

Terminating a Primary or Standby Autonomous Container Database

You can terminate primary and standby container databases separately. However, the consequences of terminating a primary container database and terminating a standby container database differ:

- Terminating a **primary container database** terminates all the primary and standby container databases. You cannot terminate a primary container database that contains Autonomous AI Database.
- Terminating a **standby container database** removes it from the Autonomous Data Guard configuration. If a primary container database no longer has any standby afterwards, it becomes a standalone container database.

If you are using a multi-standby setup with automatic failover enabled, you must disable automatic failover before terminating any standby Autonomous Container Database.

Update Autonomous Container Database Maintenance Preferences

You update the maintenance preferences of an Autonomous Container Database from its **Details** page.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the Autonomous Container Database whose maintenance schedule you want to change.

For instructions, see View Details of an Autonomous Container Database.

2. Under **Actions**, click **Edit Maintenance Preferences**.
3. You can update the following maintenance preferences from the **Edit automatic maintenance** card:
 - **Configure maintenance method:** Choose between Rolling or Non-rolling maintenance methods.
 - The **Rolling** method updates the ACD, one node at a time, with no downtime for Autonomous AI Databases.
 - The **Non-rolling** method shuts down and updates the ACD in parallel across all nodes. This method minimizes maintenance time but it requires full system downtime for the ACD and all associated Autonomous AI Databases.

Note

In an Autonomous Data Guard configuration, the non-rolling maintenance method results in downtime for the primary and standby ACDs during their respective maintenance window until the patching completes.

- Optionally, you can also select **Enable time-zone update**. Time-zone files can only be updated using the non-rolling configuration method.

Note

All the RUs that include time-zone updates will be patched using the non-rolling maintenance method (with full system downtime) only. So, if you configure your maintenance to the rolling method and enable time-zone update, all the RUs that include a time-zone update are applied in the non-rolling method only. Only those RUs without a time-zone update are applied in the rolling method.

- **Configure container database maintenance version:** Choose **Next release update (RU)** or **Latest Release update (RU)** to configure the next maintenance version of the ACD. Accordingly, the ACD will be updated to the next immediate release or the latest release in the upcoming maintenance window.
- **Configure automatic maintenance schedule:** You can change the maintenance schedule settings, as desired:
 - Select **No preference** to permit Oracle to schedule maintenance as needed based on the maintenance schedule of the Exadata Infrastructure resource hosting the Autonomous Container Database.
 - Select **Custom schedule** on Oracle Public Cloud and **Specify a schedule** on Exadata Cloud@Customer to restrict when Oracle can schedule maintenance. You can choose your preferred month, week, weekday, start time, and buffer period between primary and standby maintenance execution for the Autonomous Container Database maintenance. You must select at least one month per quarter, and you can also choose to skip patching for a quarter. Patching cannot be skipped for two consecutive quarters.

Note

When you choose to skip, you need to select at least one month from that quarter. This acts as a fallback in case maintenance did not occur in the previous un-skipped quarter. In this scenario, Oracle will automatically perform the maintenance in the selected month, even if skipping is chosen for that quarter.

Tip

For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable.

4. When finished, save your changes.

View and Manage Scheduled Maintenance of an Autonomous Container Database

You view and manage scheduled maintenance of an Autonomous Container Database from its **Details** page.

Required IAM Policies

use autonomous-container-databases

Procedure

1. Go to the **Details** page of the desired Autonomous Container Database.
For instructions, see View Details of an Autonomous Container Database.
2. Click **Maintenance**.
The **Maintenance** page lists any upcoming planned maintenance.
For more information about the details displayed in the **Maintenance** page, see Scheduled Maintenance Details.
3. For each maintenance event listed in the **Maintenance** page, you can perform the management operations listed in Management Operations on a Scheduled Maintenance, provided that the event is not already in progress.

When you reschedule an Autonomous Container Database maintenance event that is already scheduled, Oracle may place it in a queue if its Exadata Infrastructure resource or Autonomous Exadata VM Cluster resource is:

- Already undergoing a maintenance update, or
- Scheduled for a maintenance activity simultaneously as the Autonomous Container Database.

For more information about automatic queuing of maintenance events, see Automatic Queuing of Maintenance Events.

View Details of an Autonomous Container Database

You view the details of an Autonomous Container Database by going to its **Details** page.

Required IAM Policies

`inspect autonomous-container-databases`

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous Container Database**.
You can see a list of Autonomous Container Databases in your current Compartment.
3. Choose the **Compartment** containing the Autonomous Container Database whose details you want to view.
The list of Autonomous Container Databases refreshes to show those in the selected Compartment. This also includes the ACDs in the terminated state.
4. Click the name of the Autonomous Container Database whose details you want to view.
The **Details** page for the Autonomous Container Database is displayed with information about the ACD.

Features of the Details Page

You can perform a variety of actions on the **Details** page for an Autonomous Container Database, as listed below:

- View Resource Usage for an Autonomous Container Database
- Restart an Autonomous Container Database
- Move an Autonomous Container Database to a Different Compartment
- Manage Autonomous Data Guard configuration
 - Switch Roles in an Autonomous Data Guard Configuration
 - Fail Over to the Standby in an Autonomous Data Guard Configuration
 - Reinstall the Disabled Standby in an Autonomous Data Guard Configuration
 - Update Autonomous Data Guard Settings
 - Convert Physical Standby to Snapshot Standby
 - Convert Snapshot Standby to Physical Standby
- Update Autonomous Container Database Maintenance Preferences
- View and Manage Scheduled Maintenance of an Autonomous Container Database
- View Past Maintenance of an Autonomous Container Database
- You can also schedule an on-demand maintenance to update RU (Release Update) along with the time-zone file or just the time-zone file for an ACD. Refer to Schedule a Quarterly Maintenance Update for instructions.
- Rotate the Encryption Key of an Autonomous Container Database
- Edit Autonomous Container Database Backup Settings
- On Oracle Public Cloud, click **Manage tags** under **Actions**. On Exadata Cloud@Customer, click **Add tags** under **Actions**. You can view the existing tags from the **Tags** tab.
- On Oracle Public Cloud deployments, you can enable Autonomous Data Guard on an ACD that is provisioned already. Refer to Enable Autonomous Data Guard on an Autonomous Container Database for instructions.
- Create an Autonomous AI Database Software Image
- Terminate an Autonomous Container Database

Click **Autonomous AI Databases** to view a list of Autonomous AI Databases in the selected Compartment that use the chosen Autonomous Container Database. Using this list you can perform a variety of actions.

Action	Steps
Create an Autonomous AI Database	Click Create Autonomous AI Database and follow the instructions listed in Create a Dedicated Autonomous AI Database.
View details of a listed Autonomous AI Database	In the list of Autonomous AI Databases, click the name of the one whose details you want to view.
Move the listed Autonomous AI Database to a different compartment	In the list of Autonomous AI Databases, click the menu in the row of the one you want to move to a different compartment and then click Move resource .

Action	Steps
Copy the OCID of a listed Autonomous AI Database	In the list of Autonomous AI Databases, click the menu in the row of the one whose OCID you want to copy and then click Copy OCID . The OCID is copied to your clipboard and the menu item changes to Copied .
Add security attributes	APPLIES TO: Oracle Public Cloud only In the list of Autonomous AI Databases, click the menu in the row of the one where you want to add security attributes and click Add security attributes .
Add tags to a listed Autonomous AI Database	In the list of Autonomous AI Databases, click the menu in the row of the one you want to apply tags to and then click Manage tags .
Terminate a listed Autonomous AI Database	In the list of Autonomous AI Databases, click the menu in the row of the one you want to terminate and then click Terminate . Then, in the Terminate Database dialog: 1. Read the warning about terminating an Autonomous AI Database. 2. Enter the name of the Autonomous AI Database. 3. Optionally, select Delete all long-term backups that are not retention locked to remove all long-term backups of the Autonomous AI Database if they are not retention locked. 4. Submit your details to begin the termination process.

View a List of Autonomous Container Databases

You can view a list of all Autonomous Container Databases in a Compartment or of just those in a particular Autonomous Exadata VM Cluster resource.

View a List of Autonomous Container Databases in a Compartment

You view a list of all Autonomous Container Databases in a Compartment by going to the **Autonomous Container Databases** page.

Required IAM Policies

inspect autonomous-container-databases

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.

For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.

2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous Container Database**.

The **Autonomous Container Databases** page is displayed, showing the list of Autonomous Container Databases in your current Compartment. This also includes the ACDs in the terminated state. Choose the **Compartment** where you want to see a list of Autonomous Container Databases.

3. The following details are displayed for each Autonomous Container Database in the list:
 - **State** of the resource such as Updating, Provisioning, Available, etc.
 - **Availability domain** in which the resource is created (on Exadata Cloud@Customer deployments only).
 - **Compute model**, that is, ECPU or OCPU.
 - **Reclaimable CPUs** or the CPUs that continue to be included in the count of CPUs available to the Autonomous Container Database even after one of its Autonomous AI Database is terminated or scaled down. You can release these reclaimable CPUs to the available CPUs at its parent AVMC level by restarting the Autonomous Container Database. For more details, see Compute Management.
 - CPU value of the largest provisionable Autonomous AI Database.
 - **Provisioned CPUs** or the total CPUs allocated for all the Autonomous AI Databases within an Autonomous Exadata VM Cluster (AVMC).
 - If this Autonomous Container Database is a **Primary or Standby** resource in an Autonomous Data Guard configuration.
 - **Date and time of creation** of the resource.

Features of the Autonomous Container Databases Page

You can perform a variety of actions on the **Autonomous Container Databases** page.

Action	Steps
Create an Autonomous Container Database	Click Create Autonomous Container Database and follow the instructions in Create an Autonomous Container Database.
View details of a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the name of the one whose details you want to view.
Move a listed Autonomous Container Database to another compartment	In the list of Autonomous Container Databases, click the menu in the row of the one that you want to move and then click Move resource . The Move resource to a different compartment dialog opens where you can select another compartment, and click Move resource .
Copy the OCID of a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the  menu in the row of the one whose OCID you want to copy and then click Copy OCID . The OCID is copied to your clipboard and the menu item changes to Copied .
Add tags to a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the  menu in the row of the one you want to apply tags to and then click Manage tags .

Action	Steps
Restart a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the menu in the row of the one you want to restart and then click Restart . The Restart dialog appears. Type in the <i>Autonomous Container Database name</i> for confirmation, and click Restart .
Terminate a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the  menu in the row of the one you want to terminate and then click Terminate . Then, in the Terminate Autonomous Container dialog: 1. Read the warning about terminating an Autonomous Container Database. 2. Enter the name of the Autonomous Container Database. 3. Submit your details to begin the termination process.

View a List of Autonomous Container Databases in an Autonomous Exadata VM Cluster Resource

You view a list of the Autonomous Container Databases in a particular Autonomous Exadata VM Cluster resource by going to that resource's **Details** page.

Required IAM Policies

`inspect cloud-autonomous-vmclusters`

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous Exadata VM Cluster**.
The list of Autonomous Exadata VM Cluster resources in your current Compartment is displayed.
3. Choose the Compartment containing the Autonomous Exadata VM Cluster resource whose Autonomous Container Databases you want to view.
The list of Autonomous Exadata VM Cluster resources refreshes to show those in the selected Compartment.
4. Click the name of the Autonomous Exadata VM Cluster resource whose Autonomous Container Databases you want to view.

From the **Details** page for the chosen Autonomous Exadata VM Cluster resource, you can view a list of Autonomous Container Databases in the selected Compartment that use the chosen Autonomous Exadata VM Cluster resource.

Features of the List of Autonomous Container Databases

The following details are displayed for each Autonomous Container Database in the list:

- **State** of the resource such as Updating, Provisioning, Available, etc.
- **Reclaimable CPUs** or the CPUs that continue to be included in the count of CPUs available to the Autonomous Container Database even after one of its Autonomous AI Database is terminated or scaled down. You can release these reclaimable CPUs to the available CPUs at its parent AVMC level by restarting the Autonomous Container Database. For more details, see Compute Management.
- CPU value of the largest provisionable Autonomous AI Database.
- **Provisioned CPUs** or the total CPUs allocated for all the Autonomous AI Databases within an Autonomous Exadata VM Cluster (AVMC).
- If this Autonomous Container Database is a **Primary or Standby** resource in an Autonomous Data Guard configuration.
- **Date and time of creation** of the resource.

The following actions can be performed from the list of Autonomous Container Databases.

Action	Steps
Create an Autonomous Container Database	Click Create Autonomous Container Database and follow the instructions in Create an Autonomous Container Database.
View details of a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the name of the one whose details you want to view.
Move a listed Autonomous Container Database to a different compartment	In the list of Autonomous Container Database(s), click the menu in the row of the one you want to move to a different compartment, and click Move resource . The Move resource to a different compartment dialog opens where you can select another compartment, and click Move resource .
Copy the OCID of a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the  menu in the row of the one whose OCID you want to copy and then click Copy OCID . The OCID is copied to your clipboard and the menu item changes to Copied .
Add tags to a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the  menu in the row of the one you want to apply tags to and then click Manage tags .
Restart a listed Autonomous Container Database.	In the list of Autonomous Container Databases, click the menu in the row of the one you want to restart and then click Restart . The Restart dialog appears. Type in the <i>Autonomous Container Database name</i> for confirmation, and click Restart .

Action	Steps
Terminate a listed Autonomous Container Database	In the list of Autonomous Container Databases, click the  menu in the row of the one you want to terminate and then click Terminate. Then, in the Terminate Autonomous Container dialog: 1. Read the warning about terminating an Autonomous Container Database. 2. Enter the name of the Autonomous Container Database. 3. Submit the details to begin the termination process.

View NFS Space Utilization

APPLIES TO:  Exadata Cloud@Customer only

Follow these steps to view the NFS Space Utilization of an Autonomous Container Database (ACD).

Required IAM Policies

`inspect autonomous-container-databases`

Procedure:

To view the NFS space utilization for an ACD:

1. Go to the Details page of the ACD. For instructions, see View Details of an Autonomous Container Database.
2. On the Details page, locate the Backup section. If your backup destination type is NFS, you will get the percentage of space used in this section.
3. The following status icons are displayed depending on the space utilization percentage:

Table 6-2 NFS Space Utilization Status Icon Details

NFS Space Utilization percentage	Icon Used
0-70	
71-85	
>= 86	

View Past Maintenance of an Autonomous Container Database

You view past maintenance of an Autonomous Container Database from its **Details** page.

Required IAM Policies

`inspect autonomous-container-databases`

Procedure

1. Go to the **Details** page of the desired Autonomous Container Database.
For instructions, see View Details of an Autonomous Container Database.
2. Click **Maintenance History** to see a list of past maintenance.

View Resource Usage for an Autonomous Container Database

Follow these steps to view the Autonomous Container Database's resource usage details.

Required IAM Policies

inspect autonomous-container-databases

Procedure

To view the resource usage visualizations for an ACD:

1. Go to the **Details** page of the Autonomous Container Database.
For instructions, see View Details of an Autonomous Container Database.
2. On the **Details** page, locate the **Resource Usage** section. In this section, you can get a quick overview of the ACD's resource usage, as:
 - Provisioned CPUs
 - Reserved CPUs
 - Reclaimable CPUs
 - Database Memory per CPU (GB)
 - Largest provisionable ADB, that is, the maximum number of CPUs that can assigned to a new Autonomous AI Database in this ACD.

 Tip

The above resource attributes are explained in Resource Terminology.

3. To get more details and a comprehensive picture of the resource usage details for this ACD, click the **View Details** link.
4. The **Resource usage details** page opens with the following sections:
 - **CPU usage:** This section shows the number of provisioned, reserved, and reclaimable CPUs for this ACD.
 - **CPU usage at VM level:** This section has graphical and tabular representations of the ACD's resource usage.

 Tip

You can choose to see this information either in the graphical or tabular view by selecting **Chart view** or **Table view** from the drop-down list at the top-right corner of this section.

For more information on the details presented in the chart and tabular formats, refer to Resource Usage Visualizations.

Clone an Autonomous Container Database

APPLIES TO:  Oracle Public Cloud only

You can clone an Autonomous Container Database from its Details page or from a remote backup.

Clone an Autonomous Container Database

You can clone an Autonomous Container Database (ACD) from its Details page and its latest available backup will be considered as the clone source.

Prerequisites

To view the requirements for successfully cloning an ACD instance, see About Cloning Autonomous Container Database on Dedicated Exadata Infrastructure.

Required IAM Policies

manage autonomous-container-databases
use cloud-exadata-infrastructures
use cloud-autonomous-vmclusters
inspect autonomous-container-db-backups

Procedure

1. Go to the Details page of the Autonomous Container Database you want to clone.
For instructions, see View Details of an Autonomous Container Database.
2. Under More actions, click the **Create Clone** button.
3. Fill out the **Clone Autonomous Container Database** page with the following information:

Setting	Description	Notes
Compartment	Select a compartment to host the Autonomous Container Database.	
Display name	Enter a user-friendly description or other information that helps you easily identify the resource.	The display name does not have to be unique and you can change it whenever you want. Avoid entering confidential information.

Setting	Description	Notes
Container Database Name	Enter a name for the container database. It can contain only letters and numbers. Starting with a letter and a maximum of 30 characters.	The container database name must be unique across the Autonomous Exadata VM Cluster.
		 Tip The container database name is used to name the Oracle Key Vault's (OKV) wallet. You can use this name to locate the OKV wallet associated with your Autonomous Container Database on your OKV console.
Exadata Infrastructure	The Exadata Infrastructure to host the new Autonomous Container Database.	
Autonomous Exadata VM Cluster	The Autonomous Exadata VM Cluster to host the new Autonomous Container Database.	
Configure automatic maintenance	The panel to configure automatic maintenance displays the following default settings: • Maintenance method: Rolling with time-zone file update disabled. • Container database maintenance version: Next release update (RU) • Maintenance schedule: No schedule preference specified. Optionally, you can configure a maintenance preference or schedule by clicking Modify maintenance schedule that launches the Edit automatic maintenance dialog.	DST stands for Daylight Savings Time.

Setting	Description	Notes
Edit automatic maintenance	Configure maintenance method: Choose between Rolling or Non-rolling maintenance methods. Optionally, you can also select Enable time-zone update. See Service Maintenance Types for more information. Configure container database maintenance version: Choose either Next RU or Latest RU to configure the next maintenance version of the ACD. Configure automatic maintenance schedule: Optionally, you can change the maintenance schedule from the default (No preference, which permits Oracle to schedule maintenance as needed) to Custom schedule in Oracle Public Cloud and Specify a schedule in Exadata Cloud@Customer. For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable. Click Save to close this dialog.	Updating the time-zone file would require a complete downtime for the ACD and the associated Autonomous AI Databases. The downtime is dependent on the amount of data that is time-zone sensitive. All the RUs that include a time-zone update will be patched in the non-rolling maintenance method (with full system downtime) only. So, if you configure your maintenance to the rolling method and enable time-zone update, all the RUs that include a time-zone update are applied in the non-rolling method only. Only those RUs without a time-zone update are applied in the rolling method. In an Autonomous Data Guard configuration, the non-rolling maintenance method results in downtime for the primary and standby ACDs during their respective maintenance window until the patching completes.
Configure backups: Enable automatic backups	By default, automatic backups are enabled for an ACD. Optionally, you can choose to disable them by deselecting the Enable automatic backups check box. While provisioning an ACD with Autonomous Data Guard, you can not disable the automatic backups. When you enable automatic backups, choose from the following options to determine how long backups are retained after terminating the ACD: - Retain backups as per the backup retention period - Retain backups for 72hours, then delete	The setting you choose for retaining backups after ACD's termination (which is 72 hours or till the duration of the backup retention period of the ACD) does not apply to long-term backups. If disabled for an ACD, automatic backups can be enabled anytime later from the Oracle Cloud Infrastructure (OCI) console by following the steps outlined in Edit Autonomous Container Database Backup Settings. However, once enabled you can not disable automatic backups for the ACD. If enabling automatic backups fail for some reason, the ACD provisioning also fails with an error message. As a workaround, you can provision the ACD with automatic backups disabled, and enable them from the ACD's Details page later.

Setting	Description	Notes
Configure backups: Backup retention period (in days)	After enabling automatic backups, specify a Backup retention period value to meet your needs. You can choose any value between 7 to 95 days.	The backup retention policy value defaults to 15 days. All the backups are automatically deleted after the backup retention period.
Retention lock	If you choose to retain backups for the duration of the backup retention period, you can also enable retention lock. Enabling retention lock: - Provides a 14-day grace period to disable the retention lock, after which the retention lock setting becomes permanent. - Retains all the automatic database backups until the ACD backup retention period. - Retains the long-term backups until their retention period (which is set while creating a long-term backup). Allows you only to increase the retention period that is already set, guaranteeing the backups remain for the retention period. - Disallows you from deleting any backup before its retention period expires. - Deletes all the automatic backups upon reaching their retention period expiry. However, if you have configured NFS for storing the automatic backups and the underlying ACD is terminated, then you have to manually delete the backups. - Deletes the long-term backups after they expire. If you have configured NFS for storing the long-term backups and the ACD is terminated, then you have to manually delete the backups after they expire.	If you have chosen to retain backups only for 72 hours after the ACD is terminated, then the retention lock option is unavailable.
Configure backups: Enable cross-region backup copy	Optionally, select Enable cross-region backup copy and specify a region for the backup copies.	Cross region backup copy cannot be enabled on Autonomous Data Guard enabled ACDs. Cross-region backup copy cannot be enabled on Multicloud ACD resources.
Show/hide advanced options	By default, advanced options are hidden. Click Show advanced options to display them.	

Setting	Description	Notes
Advanced options: Management	Optionally, you can define a suitable value for the following resource management attributes to suit your needs: • Database split threshold (CPU): The CPU value beyond which an Autonomous AI Database will be opened across multiple nodes. The default value of this attribute is 64 for ECPUs and 16 for OCPUs. • Node failover reservation (%): Determines the percentage of CPUs reserved across nodes to support node failover. Allowed values are 0%, 25%, and 50%, with 50% being the default option. • Distribution affinity: Determines whether an Autonomous AI Database must be opened across a minimum or maximum of nodes. By default, Minimum nodes is selected with Maximum nodes being the other option. Optionally, select Enable shared server connections to support the Net services architecture.	If the Node failover reservation is set to 0%, your Autonomous AI Database may face a complete outage during VM failure and maintenance operations such as database patching and container database restart. The shared server architecture enables a database server to allow many client processes to share very few server processes, so the number of users that can be supported is increased. You cannot disable shared server architecture after provisioning the ACD. See Special-Purpose Connection Features for more details.
Advanced options: Encryption Key	Optionally, you can configure the Autonomous Container Database to use customer-managed encryption keys instead of Oracle-managed encryption keys. Select Encrypt using a customer-managed key in this tenancy and one of the following options, depending on your deployment: • OCI Vault Service: With this option, choose a Vault and a Master Encryption Key from the lists. This option is only available on Oracle Public Cloud. • Oracle Key Vault: With this option, you must choose a key store that is same as that of the source ACD. If the source ACD has an OKV endpoint group name associated, you must provide the same OKV endpoint group name here.	By default, Encrypt using Oracle-managed keys is selected. You need to create a key store to use encryption key with Oracle Key Vault. See Use Customer-Managed Keys in Oracle Key Vault for instructions. You can use customer-managed encryption keys with Autonomous Data Guard-enabled Autonomous Container Databases with the primary and standby databases located in different availability domains within the same region. For information about using customer-managed keys see About Master Encryption Keys.

Setting	Description	Notes
Advanced options: Tags	If you want to use tags, add tags by selecting a Tag Namespace , Tag Key , and Tag Value .	You can not modify the time zone settings for an Autonomous Exadata VM Cluster already provisioned. If needed, you can create a service request in My Oracle Support. For help with filing a support request, see Create a Service Request in My Oracle Support .

- Click **Clone Autonomous Container Database**.

Clone an Autonomous Container Database from a Remote Backup

You can clone an Autonomous Container Database (ACD) from its latest available remote backup.

Prerequisites

To view the requirements for successfully cloning an ACD instance, see [About Cloning Autonomous Container Database on Dedicated Exadata Infrastructure](#).

Required IAM Policies

manage autonomous-container-databases

use cloud-exadata-infrastructures

use cloud-autonomous-vmclusters

inspect autonomous-container-db-backups

Procedure

- Go to the **Remote backups** page of your compartment.
For instructions, see [View a List of Remote Backups](#).
- Click the three dots next to the remote backup that you want to clone, and then click **Create Clone**.
- Fill out the **Clone Autonomous Container Database** page with the following information:

Setting	Description	Notes
Compartment	Select a compartment to host the Autonomous Container Database.	
Display name	Enter a user-friendly description or other information that helps you easily identify the resource.	The display name does not have to be unique and you can change it whenever you want. Avoid entering confidential information.

Setting	Description	Notes
Container Database Name	Enter a name for the container database. It can contain only letters and numbers. Starting with a letter and a maximum of 30 characters.	The container database name must be unique across the Autonomous Exadata VM Cluster.  Tip The container database name is used to name the Oracle Key Vault's (OKV) wallet. You can use this name to locate the OKV wallet associated with your Autonomous Container Database on your OKV console.
Exadata Infrastructure	The Exadata Infrastructure to host the new Autonomous Container Database.	
Autonomous Exadata VM Cluster	The Autonomous Exadata VM Cluster to host the new Autonomous Container Database.	
Configure automatic maintenance	The panel to configure automatic maintenance displays the following default settings: • Maintenance method: Rolling with time-zone file update disabled. • Container database maintenance version: Next release update (RU) • Maintenance schedule: No schedule preference specified. Optionally, you can configure a maintenance preference or schedule by clicking Modify maintenance schedule that launches the Edit automatic maintenance dialog.	DST stands for Daylight Savings Time.

Setting	Description	Notes
Edit automatic maintenance	Configure maintenance method: Choose between Rolling or Non-rolling maintenance methods. Optionally, you can also select Enable time-zone update. See Service Maintenance Types for more information. Configure container database maintenance version: Choose either Next RU or Latest RU to configure the next maintenance version of the ACD. Configure automatic maintenance schedule: Optionally, you can change the maintenance schedule from the default (No preference, which permits Oracle to schedule maintenance as needed) to Custom schedule in Oracle Public Cloud and Specify a schedule in Exadata Cloud@Customer. For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable. Click Save to close this dialog.	Updating the time-zone file would require a complete downtime for the ACD and the associated Autonomous AI Databases. The downtime is dependent on the amount of data that is time-zone sensitive. All the RUs that include a time-zone update will be patched in the non-rolling maintenance method (with full system downtime) only. So, if you configure your maintenance to the rolling method and enable time-zone update, all the RUs that include a time-zone update are applied in the non-rolling method only. Only those RUs without a time-zone update are applied in the rolling method. In an Autonomous Data Guard configuration, the non-rolling maintenance method results in downtime for the primary and standby ACDs during their respective maintenance window until the patching completes.

Setting	Description	Notes
Configure backups: Enable automatic backups	By default, automatic backups are enabled for an ACD. Optionally, you can choose to disable them by deselecting the Enable automatic backups check box. While provisioning an ACD with Autonomous Data Guard, you can not disable the automatic backups. When you enable automatic backups, choose from the following options to determine how long backups are retained after terminating the ACD: - Retain backups as per the backup retention period - Retain backups for 72hours, then delete	The setting you choose for retaining backups after ACD's termination (which is 72 hours or till the duration of the backup retention period of the ACD) does not apply to long-term backups. If disabled for an ACD, automatic backups can be enabled anytime later from the Oracle Cloud Infrastructure (OCI) console by following the steps outlined in Edit Autonomous Container Database Backup Settings. However, once enabled you can not disable automatic backups for the ACD. If enabling automatic backups fail for some reason, the ACD provisioning also fails with an error message. As a workaround, you can provision the ACD with automatic backups disabled, and enable them from the ACD's Details page later.
Configure backups: Backup retention period (in days)	After enabling automatic backups, specify a Backup retention period value to meet your needs. You can choose any value between 7 to 95 days.	The backup retention policy value defaults to 15 days. All the backups are automatically deleted after the backup retention period.

Setting	Description	Notes
Retention lock	If you choose to retain backups for the duration of the backup retention period, you can also enable retention lock. Enabling retention lock: - Provides a 14-day grace period to disable the retention lock, after which the retention lock setting becomes permanent. - Retains all the automatic database backups until the ACD backup retention period. - Retains the long-term backups until their retention period (which is set while creating a long-term backup). Allows you only to increase the retention period that is already set, guaranteeing the backups remain for the retention period. - Disallows you from deleting any backup before its retention period expires. - Deletes all the automatic backups upon reaching their retention period expiry. However, if you have configured NFS for storing the automatic backups and the underlying ACD is terminated, then you have to manually delete the backups. - Deletes the long-term backups after they expire. If you have configured NFS for storing the long-term backups and the ACD is terminated, then you have to manually delete the backups after they expire.	If you have chosen to retain backups only for 72 hours after the ACD is terminated, then the retention lock option is unavailable.
Configure backups: Enable cross-region backup copy	Optionally, select Enable cross-region backup copy and specify a region for the backup copies.	Cross region backup copy cannot be enabled on Autonomous Data Guard enabled ACDs. Cross-region backup copy cannot be enabled on Multicloud ACD resources.
Show/hide advanced options	By default, advanced options are hidden. Click Show advanced options to display them.	

Setting	Description	Notes
Advanced options: Management	Optionally, you can define a suitable value for the following resource management attributes to suit your needs: • Database split threshold (CPU): The CPU value beyond which an Autonomous AI Database will be opened across multiple nodes. The default value of this attribute is 64 for ECPUs and 16 for OCPUs. • Node failover reservation (%): Determines the percentage of CPUs reserved across nodes to support node failover. Allowed values are 0%, 25%, and 50%, with 50% being the default option. • Distribution affinity: Determines whether an Autonomous AI Database must be opened across a minimum or maximum of nodes. By default, Minimum nodes is selected with Maximum nodes being the other option. Optionally, select Enable shared server connections to support the Net services architecture.	If the Node failover reservation is set to 0%, your Autonomous AI Database may face a complete outage during VM failure and maintenance operations such as database patching and container database restart. The shared server architecture enables a database server to allow many client processes to share very few server processes, so the number of users that can be supported is increased. You cannot disable shared server architecture after provisioning the ACD. See Special-Purpose Connection Features for more details.
Advanced options: Encryption Key	Optionally, you can configure the Autonomous Container Database to use customer-managed encryption keys instead of Oracle-managed encryption keys. Select Encrypt using a customer-managed key in this tenancy and one of the following options, depending on your deployment: • OCI Vault Service: With this option, choose a Vault and a Master Encryption Key from the lists. This option is only available on Oracle Public Cloud. • Oracle Key Vault: With this option, you must choose a key store that is same as that of the source ACD. If the source ACD has an OKV endpoint group name associated, you must provide the same OKV endpoint group name here.	By default, Encrypt using Oracle-managed keys is selected. You need to create a key store to use encryption key with Oracle Key Vault. See Use Customer-Managed Keys in Oracle Key Vault for instructions. You can use customer-managed encryption keys with Autonomous Data Guard-enabled Autonomous Container Databases with the primary and standby databases located in different availability domains within the same region. For information about using customer-managed keys see About Master Encryption Keys.

Setting	Description	Notes
Advanced options: Tags	If you want to use tags, add tags by selecting a Tag Namespace , Tag Key , and Tag Value .	You can not modify the time zone settings for an Autonomous Exadata VM Cluster already provisioned. If needed, you can create a service request in My Oracle Support. For help with filing a support request, see Create a Service Request in My Oracle Support .

- Click **Clone Autonomous Container Database**.

Create and Manage Autonomous AI Databases

Create an Autonomous AI Database

You can create an Autonomous AI Database on Dedicated Exadata Infrastructure from the Oracle Cloud Infrastructure console.

Note

In an Autonomous Data Guard setup, you can not create an Autonomous AI Database in an Autonomous Container Database (ACD) whose standby ACD is in a snapshot standby role.

Required IAM Policies

manage autonomous-databases

read autonomous-container-databases

Related Live Labs

For a "try it out" alternative that demonstrates these instructions, see *Lab 1: Provisioning Databases* in Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop.

Procedure

- Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see [Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console](#).
- If needed, switch to the region where you want to create the database.
See [Switching Regions](#) for information on switching regions and working in multiple regions.
- Click **Create Autonomous AI Database**.
- Fill out the **Create Autonomous AI Database** page with the following information:

Setting	Description	Notes
Compartment	Select a compartment to host the Autonomous AI Database.	
Display name	Enter a user-friendly description or other information that helps you easily identify the Autonomous AI Database.	Avoid entering confidential information.
Database name	Provide a name for the new Autonomous AI Database.	Must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous AI Databases in the same Autonomous Exadata VM Cluster.
Workload type	Choose Lakehouse or Transaction Processing .	Determines the configuration of the new Autonomous AI Database. JSON and APEX workload types are not supported on dedicated Exadata infrastructure.

Setting	Description	Notes
Autonomous Container Database	Select an Autonomous Container Database (ACD) in which to create the Autonomous AI Database. Start by choosing whether you want a standard ACD or an ACD configured to use Autonomous Data Guard. Click Change Compartment to select from a different compartment.	The Autonomous AI Database software version is determined by the Database Software Version of the ACD chosen in this step. If you choose an ACD configured to use Autonomous Data Guard, two databases of the CPU count and storage you select are created: one in a primary container database, and one (a synchronized copy) in a standby container database. For more information, see <i>Manage Primary and Standby Databases in an Autonomous Data Guard Configuration</i>. ⊗ You can not create an Autonomous AI Database in an ACD whose standby ACD is in a snapshot standby role. Suppose an ACD with a cross-region Autonomous Data Guard association has a switchover or failover, and its standby assumes the primary role. In this case, creating a Autonomous AI Database in such an ACD will fail because the replicated vault used by the standby is read-only. ⊗ You can not create an Autonomous AI Database for Developers instance on an Autonomous Data Guard enabled container database.
Configure the database: Free instance	Toggle the Free instance button on, if you want to create an Autonomous AI Database for Developers instance. ECPU count and Storage (GB) are auto-populated with 4 and 32 respectively because Autonomous AI Database for Developers comes fixed at 4 ECPUs and 32GB storage. Compute auto scaling is disabled because developer database instances do not support manual or auto-scaling.	Autonomous AI Database for Developers is a free tier offering designed for database development and functional testing activities. See <i>Autonomous AI Database for Developers</i> for more details. As developer database instances can only be created on an ECPU-based ACDs without Autonomous Data Guard, the Free instance toggle button is disabled for ACDs with OCPU, Autonomous Data Guard, or both.

Setting	Description	Notes
Configure the database: CPU count	Select the number of CPUs for your database from the list of provisionable CPUs.	The CPU type, that is, ECPU or OCPU is determined by the parent Autonomous Exadata VM Cluster's compute type. This value defaults to 4 ECPUs or 1 OCPU depending on the CPU type. Refer to Compute Models in Autonomous AI Database for more details. For databases that need 1 or more OCPUs or 2 or more ECPUs, you must specify the number of assigned CPUs as an <i>integer</i>. For example, you cannot assign 3.5 ECPUs to a database. The next available number of ECPUs above 3 is 4. You can also assign OCPU count from 0.1 to 0.9 in increments of 0.1 OCPUs for databases that do not need an entire OCPU. This allows you to overprovision CPU and run more databases on each infrastructure instance. Refer to CPU Overprovisioning for more details. ⊗ CPU overprovisioning is not supported for Autonomous AI Databases that use ECPUs.
Configure the database: CPU auto scaling	Enable or disable CPU auto scaling, which permits Autonomous AI Database to automatically use up to three times as many CPUs as specified by CPU Count as the workload on the database increases.	
Configure the database: Storage (GB)	Specify the storage to allocate to your database in terabytes (GB), with a minimum value of 32 GB.	The default values are 1024 GB for Autonomous AI Lakehouse and 32 GB for the Autonomous AI Transaction Processing workloads.

Setting	Description	Notes
Configure the database: Elastic Pools	If Compute auto scaling disabled, you can choose to create an elastic pool as a pool leader or join an existing elastic pool as a pool member. - Click Show advanced options. - Select Enable elastic pool. - To join an existing pool, select Join an existing elastic pool and select a pool leader from Select pool leader in compartment. - To create a new elastic pool, select Create an elastic pool, and select a pool size for Pool ECPU count from available list of pool shapes: 128, 256, 512, 1024, 2048, or 4096.	Only Autonomous AI Transaction Processing databases without Autonomous Data Guard that use ECPU compute model can be used to create an elastic pool.
Username	Denotes the database's username. This is a read-only value.	
Password	Set the password for the Admin database user in your new 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.
Confirm password	Re-enter the password for confirmation.	
Access Control	Optionally, click Modify Access Control to configure network access by enabling database level access control, which is disabled by default.	Clicking Modify Access Control opens the Edit Access Control List dialog.

Setting	Description	Notes
Database-level access control	To configure network access by creating an access control list (ACL), select Enable database-level access control, and click + Access Control Rule on the Edit Access Control List dialog. Depending on the types of addresses in your list, select one of the following options from the IP notation type drop-down selector: • IP Address: Specify individual IP addresses. You can use commas to separate multiple IP addresses. • CIDR Block: Specify ranges of public IP addresses using CIDR notation. You can use commas to separate multiple values.	An ACL provides additional protection to your Autonomous AI Database by allowing only the client with specific IP addresses to connect to the database. See <i>Restrict Access Using a Network Access Control List</i> for details.
Standby database access control	If the parent Autonomous Container Database uses Autonomous Data Guard, you can define access control for the standby database also. Depending on your preference, select one of the following options for Standby database access control: • Choose Same as primary database to apply the ACL that you already defined for the primary database to the standby database as well. This is the default selection. • Choose Define standby database access control to define a separate ACL for the standby database. By default, the standby database ACL is initialized with the same values as the primary database ACL. Add or modify the entries as needed.	By default, the ACL defined for the primary database applies to the standby database also.
Contact email	Provide contact emails where you can receive operational notifications, announcements, and unplanned maintenance notifications.	Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.

Setting	Description	Notes
Show/hide advanced options	By default, advanced options are hidden. Click Show advanced options to display them.	
Advanced options: Encryption key	The encryption key settings are inherited from the parent Autonomous Container Database.	
Advanced options: Management	Choose a Character Set and National Character Set from the drop-down list.	Refer to Character Set Selection for Dedicated Autonomous AI Database for more details.
Advanced options: Database In-memory	Optionally, select Enable database In-memory and adjust the percentage of System Global Area (SGA) to allocate.	Please see Database In-Memory for the requirements and guidelines to use this feature with Autonomous AI Database. If required, you can also disable Database In-Memory from the Oracle Cloud Infrastructure (OCI) console. See Enable or Disable Database In-Memory for instructions.
Advanced options: Tags	If you want to use tags, add tags by selecting a Tag Namespace , Tag Key , and Tag Value .	Tagging is a metadata system that allows you to organize and track resources within your tenancy. See Tagging Dedicated Autonomous AI Database Cloud Resources.

5. Optionally, you can save the resource configuration as a stack by clicking **Save as Stack**. You can then use the stack to create the resource through the Resource Manager service.

Enter the following details on the **Save as Stack** dialog, and click **Save**.

- **Name:** Optionally, enter a name for the stack.
- **Description:** Optionally, enter a description for this stack.
- **Save in compartment:** Select a compartment where this Stack will reside.
- **Tag namespace, Tag key, and Tag value:** Optionally, apply tags to the stack.

For requirements and recommendations for Terraform configurations used with Resource Manager, see [Terraform Configurations for Resource Manager](#). To provision the resources defined in your stack, [apply the configuration](#).

6. Submit the details to create the Autonomous AI Database.

Your display shows **Provisioning** until the new Autonomous AI Database is available.

Note

When you create an Autonomous AI Database, CPU billing is initiated, billed by the second with a minimum usage period of one minute.

Backup an Autonomous AI Database on Dedicated Exadata Infrastructure Manually

In addition to automatic backups, Autonomous AI Database on Dedicated Exadata Infrastructure also allows you to take manual backups from its **Details** page.

Note

See About Backup and Recovery for a list of supported backup types in Autonomous AI Database.

Required IAM Policies

read autonomous-databases

manage autonomous-backups

Procedure

1. Go to the **Details** page of the Autonomous AI Database you want to manually backup. For instructions, see View Details of an Autonomous AI Database on Dedicated Exadata Infrastructure.
2. Click **Backups**. The list of backups is displayed. Click **Create Manual Backup**.
3. In the **Create Manual Backup** dialog, enter a name in the **Name** field and click **Create**.

Note

While backing up a database, the database is fully functional; however during the backup life cycle management operations are not allowed. For example, stopping the database is not allowed during the backup.

Clone an Autonomous AI Database on Dedicated Exadata Infrastructure

Cloning is the process of creating a point-in-time copy of your Autonomous AI Database or its backup set. You can clone an Autonomous AI Database on Dedicated Exadata Infrastructure from its **Details** page.

Note

Autonomous AI Databases with 26ai software version can not be cloned into an Autonomous AI Database with 19c version and vice-versa.

Clone an Autonomous AI Database Instance

Prerequisites

To view the requirements for successfully cloning an Autonomous AI Database instance, see [Clone Requirements](#).

Required IAM Policies

manage autonomous-databases

read autonomous-container-databases

Procedure

Follow these steps to clone an Autonomous AI Database instance using the Oracle Cloud Infrastructure console.

1. Go to the **Details** page of the Autonomous AI Database you want to clone.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

For instructions, see [View Details of a Dedicated Autonomous AI Database](#).

2. On Oracle Public Cloud, click **Create Clone** under **More actions**, and on Exadata Cloud@Customer, click **Create Clone** under **Actions**.

Create Clone is only enabled only when the database's state is **Available**.

3. Fill out the **Create Autonomous AI Database Clone** page with the following information:

Setting	Description	Notes
Clone type	Choose Full Clone or Metadata Clone .	See Clone Types for more details.
Clone source	Select Clone from database instance .	
Autonomous AI Database clone: Compartment	Choose a compartment for the database clone.	See Understanding Compartments for information on using and managing compartments.
Autonomous AI Database clone: Source database name	This read-only field identifies the name of the source database.	
Autonomous AI Database clone: 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.

Setting	Description	Notes
Autonomous AI Database clone: Database name	Specify a name for the clone database.	Database name must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous AI Databases in the same Autonomous Exadata VM Cluster.
Region	Choose your preferred region to place the clone database.	The default is the same region as the source database. You can also perform a cross region clone and correspondingly choose another region.
Exadata Infrastructure	Specify the Exadata Infrastructure to place the clone database.	The default is the same Exadata Infrastructure as that of the source database. You can choose an Exadata Infrastructure from a different compartment by clicking Change Compartment , if necessary.
Autonomous Exadata VM Cluster	Specify the Autonomous Exadata VM Cluster (AVMC) to place the clone database. In case of Exadata Cloud@Customer, the target AVMC must have access to the source backup destination.	The default is the same AVMC resource as that of the source database. You can choose an AVMC resource from a different compartment by clicking Change Compartment , if necessary. In Exadata Cloud@Customer deployments, if the parent AVMC has the 3rd NIC enabled, all cloning operations from database instances will travel only over this 3rd NIC.
Autonomous Container Database	Specify the Autonomous Container Database (ACD) in which to place the cloned database. Start by choosing whether you want a standard ACD or an ACD configured to use Autonomous Data Guard. Click Change Compartment to select from a different compartment.	The default is the same ACD as that of the source database.

Setting	Description	Notes
Configure the database: Free instance	Toggle the Free instance button on if you want to create an Autonomous AI Database for Developers instance. The ECPU count and Storage (GB) are auto-populated with 4 and 32, respectively, because Autonomous AI Database for Developers instances come fixed at 4 ECPU and 32GB of storage. Compute auto-scaling is disabled because developer database instances do not support manual or auto-scaling.	Autonomous AI Database for Developers is a free tier offering designed for database development and functional testing activities. For more details, see Autonomous AI Database for Developers. Developer database instances can only be created on ECPU-based ACDs without Autonomous Data Guard. Therefore, the Free instance toggle button is disabled for ACDs with OCPU, Autonomous Data Guard, or both. To successfully clone an Autonomous AI Database to an Autonomous AI Database for Developers, the actual used space of the source database, rounded up to the next GB, must be <i>32GB or less</i>.
Configure the database: CPU count:	Select the number of CPUs for your clone database from the list of provisionable CPUs.	The CPU type of the clone database (OCPU or ECPU) is determined by the clone's parent Autonomous Exadata VM Cluster compute type. ⊗ You can clone an OCPU database into either OCPU or ECPU database. However, you can not clone an ECPU database into an OCPU database. This value defaults to 2 ECPU or 1 OCPU depending on the CPU type. Refer to Compute Models in Autonomous AI Database for more details. You can also assign an OCPU count from 0.1 to 0.9 in increments of 0.1 OCPUs for databases that do not need an entire OCPU. This allows you to overprovision CPU and run more databases on each infrastructure instance. Refer to CPU Overprovisioning for more details.
Configure the database: Auto scaling	Enable or disable auto scaling, which permits Autonomous AI Database to automatically use up to three times the allocated CPUs as the workload on the database increases.	

Setting	Description	Notes
Configure the database: Storage (GB)	Specify the storage you wish to make available to your database in gigabytes (GB), with a minimum value same as the source database's current storage allocation.	For a Full Clone , the minimum storage that you can specify is the source database's actual used space rounded up to the next GB.
Configure the database: Elastic Pools	If Compute auto scaling disabled, you can choose to create an elastic pool as a pool leader or join an existing elastic pool as a pool member. - Click Show advanced options. - Select Enable elastic pool. - To join an existing pool, select Join an existing elastic pool and select a pool leader from Select pool leader in compartment. - To create a new elastic pool, select Create an elastic pool, and select a pool size for Pool ECPU count from available list of pool shapes: 128, 256, 512, 1024, 2048, or 4096.	You can create an elastic pool only while cloning an Autonomous AI Transaction Processing database without Autonomous Data Guard that uses ECPU compute model.
Administrator credentials	Username: This is a read-only field auto-populated as ADMIN. Password: Set the password for the ADMIN database user in the cloned database. Confirm password: Re-enter the password for confirmation.	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. You must specify a different password from the one for the ADMIN database user in the source database; otherwise, the clone operation will fail.
Configure network access	To enable database-level access control, click Modify Access Control and add or modify entries, as applicable.	See Set Access Control List for a Dedicated Autonomous AI Database for details.
Contact email	Provide contact emails where you can receive operational notifications, announcements, and unplanned maintenance notifications.	Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.
Show/hide advanced options	By default, advanced options are hidden. Click Show advanced options to display them.	

Setting	Description	Notes
Advanced options: Encryption key	The Encryption Key settings apply from the target Autonomous Container Database.	The source and target can be different keystore types for encryption key. However, if both use customer managed keys using Oracle Key Vault (OKV), they must use the same OKV destination. If the source database is using customer managed keys in OKV, the clone database's ACD must use the same key store and OKV endpoint group name as that of the source.
Advanced options: Management	The clone database inherits its Character Set and National Character Set values from its source.	
Advanced options: Database In-memory	Optionally, select Enable database In-memory and adjust the percentage of System Global Area (SGA) to allocate.	Please see Database In-Memory for the requirements and guidelines to use this feature with Autonomous AI Database. If required, you can also disable Database In-Memory from the Oracle Cloud Infrastructure (OCI) console. See Enable or Disable Database In-Memory for instructions.
Advanced options: Tags	If you want to use tags, add tags by selecting a Tag Namespace , Tag Key , and Tag Value .	Tagging is a metadata system that allows you to organize and track resources within your tenancy. See Tagging Dedicated Autonomous AI Database Cloud Resources.

4. Click **Clone**.

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

Clone Autonomous AI Database from Backup

Prerequisites

To view the requirements for successfully cloning an Autonomous AI Database instance, see Clone Requirements.

Required IAM Policies

manage autonomous-databases

read autonomous-container-databases

Procedure

Follow these steps to clone an Autonomous AI Database from backup using the Oracle Cloud Infrastructure console.

1. Go to the **Details** page of the Autonomous AI Database you want to clone.
For instructions, see [View Details of a Dedicated Autonomous AI Database](#).
2. There are two ways to open the **Create Autonomous AI Database Clone** page from the **Details** page:

- **Option 1:** On Oracle Public Cloud, click **Create Clone** under **More actions**, and on Exadata Cloud@Customer, click **Create Clone** under **Actions**.

When you open the **Create Autonomous AI Database Clone** page using this option, you see additional options on this page that allows you to select a specific backup for cloning.

Note

Create Clone is enabled only when the database's state is **Available**.

- **Option 2:** Click **Backups**. The list of backups is displayed. In the desired row, click , and click **Create Clone** for the backup to be cloned.

When you open the **Create Autonomous AI Database Clone** page using this option, the page launches with preselected backup for cloning. You do not see an option to choose a backup on this page again.

3. Fill out the **Create Autonomous AI Database Clone** page with the following information:

Setting	Description	Notes
Clone type	Choose Full Clone .	Metadata Clone is not supported when cloning from an Autonomous AI Database backup. See Clone Types for more details.
Clone source	Select Clone from a backup .	When you open the Create Autonomous AI Database Clone page from the  menu of a specific backup, this page launches with preselected backup for cloning. You do not see an option to choose a backup on this page again.

Setting	Description	Notes
Choose a specific backup	Once you select Clone from a backup, you see three additional options to choose a specific backup. Point in time clone: Select this option if you want to clone a backup with a specific timestamp. Selecting this option requires you to enter the backup timestamp in Enter Timestamp. You must enter a timestamp within the set retention period of the Autonomous Container Database. Select the backup from a list: Select this option if you want to choose a backup from a list of backups. You can limit the number of backups you see by specifying a period using the From and To calendar fields. Latest backup timestamp: Select this option if you want to clone a backup with the latest timestamp. Selecting the Latest backup timestamp clone type will clone the database from the most recent backup data that is available.	Point in time clone is the default option. You can not use this option to clone a long-term backup. You must choose Select the backup from a list option if you want to clone from a long-term backup. You can only select <i>one</i> backup from the list that displays based on your input date range.
Compartment	Choose a compartment for the database clone.	See Understanding Compartments for information on using and managing compartments.
Source database name	This read-only field identifies the name of the source database.	
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 a name for the clone database.	Database name must consist of letters and numbers only. The maximum length is 30 characters. The same database name cannot be used for multiple Autonomous AI Databases in the same Autonomous Exadata VM Cluster.
Region	Choose your preferred region to place the clone database.	The default is the same region as the source database. You can also perform a cross region clone and correspondingly choose another region.

Setting	Description	Notes
Exadata Infrastructure	Specify the Exadata Infrastructure to place the clone database.	The default is the same Exadata Infrastructure as that of the source database. You can choose an Exadata Infrastructure from a different compartment by clicking Change Compartment, if necessary.
Autonomous Exadata VM Cluster	Specify the Autonomous Exadata VM Cluster (AVMC) to place the clone database. In case of Exadata Cloud@Customer, the target AVMC must have access to the source backup destination.	The default is the same AVMC resource as that of the source database. You can choose an AVMC resource from a different compartment by clicking Change Compartment, if necessary. In Exadata Cloud@Customer deployments, if the parent AVMC has the 3rd NIC enabled, all cloning operations from database instances will travel only over this 3rd NIC.
Autonomous Container Database	Specify the Autonomous Container Database (ACD) in which to place the cloned database. Start by choosing whether you want a standard ACD or an ACD configured to use Autonomous Data Guard. Click Change Compartment to select from a different compartment.	The default is the same Autonomous Container Database as that of the source database.

Setting	Description	Notes
Configure the database: Free instance	Toggle the Free instance button on if you want to create an Autonomous AI Database for Developers instance. The ECPU count and Storage (GB) are auto-populated with 4 and 32, respectively, because Autonomous AI Database for Developers instances come fixed at 4 ECPUs and 32GB of storage. Compute auto-scaling is disabled because developer database instances do not support manual or auto-scaling.	Autonomous AI Database for Developers is a free tier offering designed for database development and functional testing activities. For more details, see Autonomous AI Database for Developers. Developer database instances can only be created on ECPU-based ACDs without Autonomous Data Guard. Therefore, the Free instance toggle button is disabled for ACDs with OCPU, Autonomous Data Guard, or both. To successfully clone a backup to an Autonomous AI Database for Developers, the source database's allocated space must be 32GB or less. If this condition is not met, you can attempt to clone the <i>database instance</i> to a developer database instead, provided that its <i>actual used space</i>, rounded up to the next GB, is 32GB or less.

Setting	Description	Notes
Configure the database: CPU count	Select the number of CPUs for your clone database from the list of provisionable CPUs. After the clone, you can resize to a lower value if needed. In case of OCPUs, you can even resize the CPU count to less than 1 OCPU (0.1 to 0.9 in increments of 0.1 OCPUs).	There is a minimum requirement of 2 OCPUs or 8 ECPUs for an Autonomous AI Database clone from Backup. The CPU type of the clone database (OCPU or ECPU) is determined by the clone's parent Autonomous Exadata VM Cluster compute type. Refer to Compute Models in Autonomous AI Database for more details. ⊗ You can clone an OCPU database into either OCPU or ECPU database. However, you can not clone an ECPU database into an OCPU database. The total number of CPUs available to all databases within the Autonomous Exadata VM Cluster depends on the infrastructure shape and what is already allocated to other Autonomous AI Databases. The time taken to clone an Autonomous AI Database depends on the CPU Count and the network bandwidth between the Backup Destination and the target Autonomous Container Database. You can also assign an OCPU count from 0.1 to 0.9 in increments of 0.1 OCPUs for databases that do not need an entire OCPU. This allows you to overprovision CPU and run more databases on each infrastructure instance. Refer to CPU Overprovisioning for more details.
Configure the database: Storage (GB)	Specify the storage you wish to make available to your database in gigabytes (GB), and it depends on the storage available to use.	The minimum storage that you can specify is the source database's current storage allocation.

Setting	Description	Notes
Configure the database: Elastic Pools	If Compute auto scaling disabled, you can choose to create an elastic pool as a pool leader or join an existing elastic pool as a pool member. - Click Show advanced options. - Select Enable elastic pool. - To join an existing pool, select Join an existing elastic pool and select a pool leader from Select pool leader in compartment. - To create a new elastic pool, select Create an elastic pool, and select a pool size for Pool ECPU count from available list of pool shapes: 128, 256, 512, 1024, 2048, or 4096.	-
Configure the database: Auto scaling	Enable or disable auto scaling, which permits Autonomous AI Database to automatically use up to three times the allocated CPUs as the workload on the database increases.	
Administrator credentials	Username: This is a read-only field auto-populated as ADMIN. Password: Set the password for the ADMIN database user in the cloned database. Confirm password: Re-enter the password for confirmation.	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. You must specify a different password from the one for the ADMIN database user in the source database; otherwise, the clone operation will fail.
Configure network access	To enable database-level access control, click Modify Access Control and add or modify entries, as applicable.	See Set Access Control List for a Dedicated Autonomous AI Database for details.
Contact email	Provide contact emails where you can receive operational notifications, announcements, and unplanned maintenance notifications.	Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.
Show/hide advanced options	By default, advanced options are hidden. Click Show advanced options to display them.	

Setting	Description	Notes
Advanced options: Encryption key	The Encryption Key settings apply from the target Autonomous Container Database.	The source and target can be different keystore types for encryption key. However, if both use customer managed keys using Oracle Key Vault (OKV), they must use the same OKV destination. If the source database is using customer managed keys in OKV, the clone database's ACD must use the same key store and OKV endpoint group name as that of the source.
Advanced options: Database In-memory	Optionally, select Enable database In-memory and adjust the percentage of System Global Area (SGA) to allocate.	Please see Database In-Memory for the requirements and guidelines to use this feature with Autonomous AI Database. If required, you can also disable Database In-Memory from the Oracle Cloud Infrastructure (OCI) console. See Enable or Disable Database In-Memory for instructions.
Advanced options: Tags	If you want to use tags, add tags by selecting a Tag Namespace , Tag Key , and Tag Value .	Tagging is a metadata system that allows you to organize and track resources within your tenancy. See Tagging Dedicated Autonomous AI Database Cloud Resources.

4. Submit the page to create the clone.

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

Cross Tenancy Clone an Autonomous AI Database Instance

APPLIES TO:  Oracle Public Cloud only

You can clone an Autonomous AI Database instance from one tenancy (source tenancy) to a different tenancy (destination tenancy).

The steps demonstrated in this topic apply to cross tenancy cloning with the source database and the cloned database in the same region or different regions (cross-region).

Note

Cross tenancy cloning option is only available using the CLI or the Autonomous AI Database REST APIs. You cannot create a cross tenancy clone using the Oracle Cloud Infrastructure Console.

To create a cross tenancy clone from an Autonomous AI Database instance, you must first ensure all the clone requirements are met. Specifically, you need to define the OCI Identity and

Access Management policies to authorize cross tenancy cloning. See Cross Tenancy Clone Requirements for details.

On the tenancy where you want to create the clone, that is, on the destination tenancy in the destination region, use the **CLI** or call the **REST API**, specifying a valid clone type (Full or Metadata) and the OCID of the source database, which resides in a different tenancy (the source tenancy).

-
- [CLI](#)
 - [REST API](#)

CLI

```
oci db autonomous-database create-from-clone
  --clone-type metadata
  --compartment-id ocid1.compartment.oc1..unique_ID
  --source-id ocid1.autonomoustatabase.oc1..unique_ID
  --db-name dbnameclone
  --display-name dbnameclone
  --admin-password <password>
  --data-storage-size-in-tbs 1
  --compute-model ECPU
  --compute-count 4
  --autonomous-container-database-id
  ocid1.autonomouscontainerdatabase.oc1..unique_ID
```

In the above example:

- `source-id` is the OCID of the source Autonomous AI Database.
- `autonomous-container-database-id` is the OCID of the destination Autonomous Container Database.
- `compartment-id` is OCID of the compartment at the destination where the clone database would reside.

Once the command succeeds a work-request-id will be returned which can be used to track the progress of the clone. See [create-from-clone](#) for more information.

For information about SDKs, see [Software Development Kits and Command Line Interface](#).

REST API

To create a cross tenancy clone from an Autonomous AI Database instance using REST API, use the CreateAutonomousDatabase API.

```
oci raw-request
  --debug
  --http-method POST
  --target-uri https://database.us-ashburn-1.oraclecloud.com/20160918/
  autonomousDatabases
  --request-body '{
    "cloneType": "full",
    "displayName": "RAcloneV2",
    "dbName": "RAcloneV2",
```

```
"compartmentId": "ocid1.compartment.oc1..unique_ID",
"sourceId": "ocid1.autonomousdatabase.oc1..unique_ID",
"autonomousContainerDatabaseId":
"ocid1.autonomouscontainerdatabase.oc1..unique_ID",
"adminPassword": "<password>",
"dataStorageSizeInTBs": 1,
"computeCount": 1,
"computeModel": "ECPU",
"isDedicated": "True"
}'
```

Note

Ensure to set `isDedicated` to `True` to clone an Autonomous AI Database on Dedicated Exadata Infrastructure.

See [CreateAutonomousDatabase](#) and [CreateAutonomousDatabaseCloneDetails](#) for additional information on these REST API.

For information about using the API and signing requests, see [REST APIs](#) and [Security Credentials](#).

Cross Tenancy Clone an Autonomous AI Database from Backup

APPLIES TO:  Oracle Public Cloud only

You can clone an Autonomous AI Database backup from one tenancy (source tenancy) to a different tenancy (destination tenancy).

This topic demonstrates the steps to create a cross tenancy clone from a backup.

Note

Cross tenancy cloning option is only available using the CLI or the Autonomous AI Database REST APIs. You cannot create a cross tenancy clone using the Oracle Cloud Infrastructure Console.

To create a cross tenancy clone from an Autonomous AI Database backup, you must first ensure all the clone requirements are met. Specifically, you need to define the OCI Identity and Access Management policies to authorize cross tenancy cloning. See Cross Tenancy Clone Requirements for details.

On the tenancy where you want to create the clone, that is, on the destination tenancy in the destination region, use the **CLI** or call the **REST API**, specifying a valid clone type (Full or Metadata) and the OCID of the source database, which resides in a different tenancy (the source tenancy).

- [CLI](#)

- [REST API](#)

CLI

```
oci db autonomous-database create-from-backup-timestamp
  --compartment-id ocid1.compartment.oc1..unique_ID
  --autonomous-database-id ocid1.autonomoustatabase.oc1..unique_ID
  --clone-type FULL
  --timestamp 2024-04-16T20:30:00.123456Z
  --db-name ZZC08X42GU0B09GECA
  --display-name clonedb
  --admin-password <password>
  --data-storage-size-in-tbs 1
  --compute-model ECPU
  --compute-count 4
  --autonomous-container-database-id
ocid1.autonomouscontainerdatabase.oc1..unique_ID
```

In the above example:

- autonomous-database-id is the OCID of the source Autonomous AI Database backup.
- autonomous-container-database-id is the OCID of the destination Autonomous Container Database.
- compartment-id is OCID of the compartment at the destination where the clone database would reside.

Once the command succeeds a work-request-id will be returned which can be used to track the progress of the clone. See [create-from-backup-timestamp](#) and [create-from-backup-id](#) for more information.

For information about SDKs, see [Software Development Kits and Command Line Interface](#).

REST API

To create a cross tenancy clone from a backup of an existing Autonomous AI Database using REST API, use the CreateAutonomousDatabase API.

Example 1: Cross tenancy cloning using the backup ID.

```
oci raw-request
  --debug
  --http-method POST
  --target-uri https://database.us-ashburn-1.oraclecloud.com/20160918/
autonomousDatabases
  --request-body '{
    "compartmentId": "ocid1.compartment.oc1..unique_ID",
    "dbName": "CTRAclonefrombackupid",
    "autonomousDatabaseBackupId":
"ocid1.autonomoustatabasebackup.oc1..unique_ID",
    "cloneType": "FULL",
    "cpuCoreCount": 1,
    "dataStorageSizeInTBs": 1,
    "adminPassword": "<password>",
    "displayName": "CTRAclonefrombackupid",
    "autonomousContainerDatabaseId":
"ocid1.autonomouscontainerdatabase.oc1..unique_ID",
```

```
"source": "<BACKUP_FROM_ID>"
}'
```

Example 2: Cross tenancy cloning using the backup timestamp.

```
oci raw-request
--debug
--http-method POST
--target-uri https://database.us-ashburn-1.oraclecloud.com/20160918/
autonomousDatabases
--request-body '{
  "compartmentId": "ocid1.compartment.oc1..unique_ID",
  "dbName": "CTbackuptimestamopRA",
  "autonomousDatabaseId": "ocid1.autonomousdatabase.oc1..unique_ID",
  "timestamp": "2024-07-17T02:20:15.123456Z",
  "cloneType": "FULL",
  "cpuCoreCount": 1,
  "dataStorageSizeInTBs": 1,
  "adminPassword": "<password>",
  "displayName": "CTbackuptimestamopRA",
  "autonomousContainerDatabaseId":
"ocid1.autonomouscontainerdatabase..unique_ID",
  "source": "<BACKUP_FROM_TIMESTAMP>"
}'
```

See [CreateAutonomousDatabase](#), [CreateAutonomousDatabaseFromBackupDetails Reference](#), and [CreateAutonomousDatabaseFromBackupTimestampDetails Reference](#) for additional information on these REST API.

For information about using the API and signing requests, see [REST APIs](#) and [Security Credentials](#).

Create a Long-Term Backup

In addition to automatic backups, Autonomous AI Database on Dedicated Exadata Infrastructure also allows you to take manual backups that can be retained for a minimum of 90 days and a maximum of 10 years.

Note

Long-term backups are not available with Autonomous AI Database for Developers instances. See Autonomous AI Database for Developers for more details. Additionally, long-term backups are not available unless automatic backups are enabled for the parent ACD.

Procedure

You can create a long-term backup from the Details page of an Autonomous AI Database.

Required IAM Policies

Oracle Public Cloud Deployments:

- `read autonomous-databases`
- `manage autonomous-backups`

Exadata Cloud@Customer Deployments:

- `read autonomous-databases`
- `manage autonomous-backups`
- `inspect backup-destinations`

Steps

1. Go to the **Details** page of the Autonomous AI Database for which you want to create a long-term backup.

For instructions, see [View Details of an Autonomous AI Database on Dedicated Exadata Infrastructure](#).

Note

In an Autonomous Data Guard setup, you can initiate a long-term backup from the primary or standby database. However, the long-term backup is created only where it is initiated. Suppose you want long-term backups for both the primary and standby databases. In that case, you have to initiate two long-term backups, one from the primary and the other from the standby database. However, the long-term backups created in either a primary or a standby database will be visible in all Autonomous Data Guard regions.

2. Click **Backups**. The list of backups is displayed.
3. Click **Create long-term backup**.
4. Provide the following details in the **Create long-term backup** dialog:

Setting	Description	Notes
Name	A user-friendly description or other information that helps you easily identify the backup.	Must be 1 to 30 characters, of letters, numbers, underscores (_) and hyphens (-). Must start with a letter or underscore(_), cannot contain two successive hyphens(--), or cannot end with a hyphen(-).

Setting	Description	Notes
Backup Destination	Backup destination type: This is defaulted to Network File System (NFS) and can not be modified. Backup destination: Select a backup destination from the list of backup destinations, changing compartments if necessary.	APPLIES TO:  Exadata Cloud@Customer only Irrespective of the parent container database's backup destination type, your long-term backup's destination remains NFS. To learn more about configuring NFS backup destination for Cloud@Customer, see Prerequisites for Backup Destinations for Exadata Database Service on Cloud@Customer If you choose NFS as a backup destination type while creating a long-term backup, then you have to manually delete the backups after they expire if the corresponding ACD is already terminated. See Backup Retention Policy for more details.
 Note On Oracle Public Cloud deployments, the Long-term backups are always created in Object Storage Service, irrespective of the backup destination chosen while creating the corresponding ACD.		
Retention Lock	Displays if the retention lock has been enabled for the parent ACD. The values are: Yes - The retention lock is enabled for the parent ACD No - The retention lock is not enabled for the parent ACD	The retention lock setting is inherited from the parent ACD and cannot be edited or changed. If the retention lock is enabled for the parent ACD, then the long-term backups cannot be deleted until the end of their retention period.
Retention Period	Set the retention period for the long-term backup.	The minimum allowed value for retention period is 90 days and the maximum 10 years.

5. Click **Create**.

Note

While backing up a database, the database is fully functional; however during the backup, life cycle management operations are not allowed. For example, stopping the database is not allowed during the backup.
You can see all the long-term backups of an Autonomous AI Database under Autonomous standalone backups on the OCI console.

Enable or Disable Auto Scaling of an Autonomous AI Database

Auto scaling enables a database to use up to three times more CPU and IO resources than its specified CPU count. When auto scaling is enabled, if your workload requires additional CPU and IO resources the database automatically uses the resources without any manual intervention required.

Required IAM Permissions

use autonomous-databases

Procedure

1. Go to the **Details** page of the Autonomous AI Database for which you want to enable or disable auto scaling.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

For instructions, see View Details of an Autonomous AI Database on Dedicated Exadata Infrastructure.

2. On Oracle Public Cloud, click **Edit** in the **Resource Allocation** section. On Exadata Cloud@Customer, click **Edit** in the **Resources** section. The **Manage scaling** page opens.
3. To enable auto scaling, select the check-box next to **Compute auto scaling** for **ECPU count** or **OCPU count**. If this check-box is pre-selected, deselecting it will disable auto scaling.

Tip

The CPU type, that is, ECPU or OCPU is determined by the compute model of the parent Autonomous Exadata VM Cluster. See Compute Models in Autonomous AI Database for more details.

Note

Alternatively, you can open **Manage scaling** page by clicking **Manage resource allocation** under **More actions** on Oracle Public Cloud or **Manage scaling** under **Actions** on Exadata Cloud@Customer.

4. Submit the details and apply your changes.

✓ **Tip**

Enter the following SQL commands to view:

- The number of ECPU's the database is currently using:

```
SELECT AVG_RUNNING_SESSIONS FROM V$SRCPDBMETRIC;
```

- The number of OCPU's the database is currently using:

```
SELECT AVG_RUNNING_SESSIONS/2 FROM V$SRCPDBMETRIC;
```

Enable or Disable Database In-Memory

You can enable or disable Database In-Memory from the **Details** page of an Autonomous AI Database.

Please see Database In-Memory for the requirements and guidelines to use this feature with Autonomous AI Database.

Required IAM Permissions

use autonomous-databases

Procedure

1. Go to the **Details** page of the database for which you want to enable or disable Database In-Memory. For instructions, see View Details of a Dedicated Autonomous AI Database

📘 **Note**

For databases that use Autonomous Data Guard, go to the **Details** page of the primary database.

2. On Oracle Public Cloud, click **Edit** in the **Resource Allocation** section. On Exadata Cloud@Customer, click **Edit** in the **Resources** section. The **Manage scaling** page opens.
3. To enable Database In-Memory, select **Enable database In-memory**. If this check box is pre-selected, deselecting it will disable Database In-Memory.

📘 **Note**

Alternatively, you can open **Manage scaling** page by clicking **Manage resource allocation** under **More actions** on Oracle Public Cloud or **Manage scaling** under **Actions** on Exadata Cloud@Customer.

4. When you enable Database In-Memory, adjust the slider to set a percentage of System Global Area (SGA) to the In-Memory area.
5. Submit the details and apply your changes.

Enable Ops Insights for an Autonomous AI Database

APPLIES TO:  Oracle Public Cloud only

Autonomous AI Database includes Ops Insights as one of its associated services. Ops Insights is a cloud-native service that enables users to make informed, data-driven, Autonomous AI Database resource and performance management decisions.

Required IAM Policies

use database-family

See [Set Up Groups, Users and Policies](#) for more details.

Procedure

By default, Ops Insights is disabled for an Autonomous AI Database, and you must enable it from the Oracle Cloud Infrastructure console.

To enable Ops Insights for your database:

1. Go to the **Details** page of the Autonomous AI Database.

For instructions, see [View Details of a Dedicated Autonomous AI Database](#).

Note

For dedicated databases that use Autonomous Data Guard, go to the **Details** page of the primary database. Ops Insights is not supported on a Standby Autonomous AI Database.

2. On the **Details** page, locate **Ops Insights** under **Associated Services**.
3. Click the **Enable** link next to Ops Insights.

Note

For an Autonomous AI Database with Ops Insights already enabled, you would see the **Disable** link instead. Click this link to disable Ops Insights for your database.

4. On the confirmation dialog, click **Enable**.

You see **Enabling** next to Ops Insights. After the Ops Insights is enabled, a link to view the operations insights for your database appears. Refer to [Use Ops Insights on Dedicated Autonomous AI Database](#) for more details.

View Details of an Autonomous AI Database

You view the details of an Autonomous AI Database by going to its **Details** page.

Required IAM Policies

inspect autonomous-databases

Procedure

1. Sign in to your Oracle Cloud Account at cloud.oracle.com.
2. Click the  menu icon in the top left corner to display the navigation menu.
3. Click **Oracle Database** in the navigation menu.
4. Choose an item under Oracle Database based on whether your dedicated infrastructure resources are on Oracle Cloud or Oracle Exadata Cloud@Customer:
 - On **Oracle Public Cloud**, click **Autonomous AI Database on Dedicated Infrastructure**.
The **Autonomous AI Databases** page opens, showing the list of databases in your current Compartment.
 - On **Oracle Exadata Cloud@Customer**, click **Oracle Exadata Database Service on Cloud@Customer**.
The **VM Clusters** page opens. On this page, click the type of Autonomous AI Database resource you want to manage.
5. If needed, switch to the region hosting the database.
See [Switching Regions](#) for information on switching regions and working in multiple regions.
6. If needed, switch to the compartment hosting the database.
See [Understanding Compartments](#) for information on using and managing compartments.
7. You can optionally include the list of Autonomous AI Database in the terminated state. Choose **Terminated Databases** in the search field, select **Include** and click **Apply Filter**.
8. In the list of Autonomous AI Databases, select the name of the Autonomous AI Database you want.
The **Details** page for the selected Autonomous AI Database is displayed.

Manage Customer Contacts for an Autonomous AI Database

You can add, edit, and remove customer contacts for an Autonomous AI Database from its **Details** page.

Required IAM Policies

use autonomous-databases

Procedure

1. Go to the **Details** page of the database for which you want manage customer contacts. For instructions, see [View Details of an Autonomous AI Database on Dedicated Exadata Infrastructure](#).

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the primary database.

2. On Oracle Public Cloud, click **Manage customer contacts** under **More actions**, and on Exadata Cloud@Customer, click **Manage customer contacts** under **Actions**.

Note

The lifecycle state of the Autonomous AI Database will change to **Updating** while the customer contact list is updated after you add, edit, or remove contacts.

The **Manage customer contacts** page opens, enabling you to perform the following actions:

Add Customer Contacts

You can provide up to ten contacts' email address for operational notifications and announcements. In Oracle Public Cloud, click **Add Contact** after entering each email address. In Exadata Cloud@Customer, click **+Another Contact** after entering details for one contact. Click **Save** to save your changes.

Note

Oracle recommends using the email address of an administrator group rather than an individual's, whenever possible, to ensure no important notifications or announcements are missed.

Edit Customer Contacts

Select the email addresses that you are editing. Click **Save** after making the changes.

Delete Customer Contacts

Select the contact detail to be removed. Click **X** at the end of the row. The contact details are removed. Save your changes.

Manage CPU or Storage Resources of an Autonomous AI Database

You can manage CPU or storage resources of an Autonomous AI Database from its **Details** page.

Note

In an Autonomous Data Guard setup, you can not scale an Autonomous AI Database whose standby database is in snapshot standby role. You must convert the standby Autonomous Container Database (ACD) to physical standby role to scale this database. See *Convert Snapshot Standby to Physical Standby* for instructions.

Add CPU or Storage Resources to an Autonomous AI Database

Required IAM Policies

use autonomous-databases

Related Live Labs

For a "try it out" alternative that demonstrates these instructions, see *Lab 11: Zero Downtime Scaling* in [Oracle Autonomous AI Database Dedicated for Developers and Database Users Workshop](#).

Procedure

1. Go to the **Details** page of the Autonomous AI Database you want to add CPU or storage resources to.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

For instructions, see [View Details of an Autonomous AI Database on Dedicated Exadata Infrastructure](#).

2. On Oracle Public Cloud, click **Edit** in the **Resource Allocation** section. On Exadata Cloud@Customer, click **Edit** in the **Resources** section. The **Manage scaling** page opens.
 - ⊗ This option is not enabled for an Autonomous AI Database for Developers instance.

Note

Alternatively, you can open **Manage scaling** page by clicking **Manage resource allocation** under **More actions** on Oracle Public Cloud or **Manage scaling** under **Actions** on Exadata Cloud@Customer.

3. Select the change in resources for your scale request:
 - Click the up arrow to select a value for **ECPUs count** or **OCPUs count**. The default is no change.

For databases using ECPUs, you must increment the number of assigned ECPUs to an integer. For example, you cannot assign 3.5 ECPUs to a database. The next available number of ECPUs above 3 is 4.

For databases using OCPUs that do not need an entire OCPU, you can increment the OCPU count from 0.1 to 0.9 (in increments of 0.1). This is called CPU overprovisioning. Refer to [CPU Overprovisioning](#) for requirements and limitations of CPU overprovisioning.

Note

Scaling up a database with CPU overprovisioning to use a full OCPU does not impact the predefined database services that you can connect to. That is, you can *only* connect to the **tp** and **low** services for the Autonomous AI Transaction Processing workloads and to the **low** services for the Autonomous AI Lakehouse workloads. See Predefined Database Service Names for Autonomous AI Databases to view a list of predefined services supported by an Autonomous AI Database.

For databases using 1 or more OCPUs, you must increment the number of assigned OCPUs by an integer. For example, you cannot assign 3.5 OCPUs to a database. The next available number of OCPUs above 3 is 4.

The selected CPU count is validated against a list of provisionable CPUs, and if the database can not be scaled up to the chosen CPU count, you will be provided with the two nearest provisionable CPU values. For more details about provisionable CPUs, see How VM Cluster Nodes Affect CPU Management.

Tip

You can use the [GetAutonomousDatabase](#) API to get a complete list of provisionable CPU values.

- Click up arrow to select a value for **Storage (GB)**. Alternatively, you can also enter the value directly. The default is no change.
4. Click **Update** to change your resources.

Remove CPU or Storage Resources from an Autonomous AI Database

Required IAM Policies

use autonomous-databases

Procedure

1. Go to the **Details** page of the Autonomous AI Database you want to remove CPU or storage resources from.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

For instructions, see View Details of an Autonomous AI Database on Dedicated Exadata Infrastructure.

2. On Oracle Public Cloud, click **Edit** in the **Resource Allocation** section. On Exadata Cloud@Customer, click **Edit** in the **Resources** section. The **Manage scaling** page opens.
 - ⊗ This option is not enabled for an Autonomous AI Database for Developers instance.

Note

Alternatively, you can open **Manage scaling** page by clicking **Manage resource allocation** under **More actions** on Oracle Public Cloud or **Manage scaling** under **Actions** on Exadata Cloud@Customer.

3. Select the change in resources for your scale request:

- Click the down arrow to select a value for **ECPU count** or **OCPU count**. The default is no change.

For databases using ECPUs, you must decrement the number of assigned ECPUs to an integer. For example, you cannot assign 3.5 ECPUs to a database. The next available number of ECPUs below 4 is 3. You can not scale down the ECPUs to a value less than 2.

For databases using less than 1 OCPU, you can decrement the OCPU value from 0.9 to 0.1 (in decrements of 0.1) for OCPUs. Refer to CPU Overprovisioning for requirements and limitations of CPU overprovisioning.

Note

Scaling down a database to use CPU over-provisioning (OCPU value less than 1) from a full CPU (positive integer) does not impact the predefined database services that you can connect to. Despite being on CPU over-provisioning, you can continue to connect to all the predefined database services, as done before scaling down. See Predefined Database Service Names for Autonomous AI Databases to view a list of predefined services supported by an Autonomous AI Database.

For databases using more than 1 OCPU, you must decrement the number of assigned OCPUs by an integer. For example, you cannot assign 3.5 OCPUs to a database. The next available number of OCPUs below 4 is 3.

The selected CPU count is validated against a list of provisionable CPUs, and if the database can not be scaled down to the chosen CPU count, you will be provided with the two nearest provisionable CPU values. For more details about provisionable CPUs, see How VM Cluster Nodes Affect CPU Management.

Tip

You can use the [GetAutonomousDatabase](#) API to get a complete list of provisionable CPU values.

- Click down arrow to select a value for **Storage (GB)**. The default is no change. The minimum storage that you can specify is the source database's actual used space rounded up to the next GB.

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

Manage Long-Term Backups

You can view the list of long-term backups for an Autonomous AI Database on Dedicated Exadata Infrastructure and modify or delete them from its **Details** page.

Required IAM Policies

- read autonomous-databases
- manage autonomous-backups

Procedure

1. Go to the **Details** page of the Autonomous AI Database for which you want to modify a long-term backup.

For instructions, see View Details of an Autonomous AI Database on Dedicated Exadata Infrastructure.

2. Click **Backups**.

All the existing backups for this Autonomous AI Database are listed with the details such as its name, state, type, retention period, Database version, backup destination, expiration date, and encryption key.

3. Long-term backups are listed with **Long-term** as their **Type**. You can perform the following actions on a long-term backup by clicking the  menu next to it.

Action	Steps
Clone a long-term backup	Click the actions icon, and select Create Clone. The Clone Autonomous AI Database page opens with Clone Source defaulted to this long-term backup. Provide the remaining details on this page, and click Clone Autonomous AI Database. Refer to Clone Autonomous AI Database from Backup for more details.
Edit the retention period	Click the actions icon, and select Edit retention period. Choose the Retention period of your choice from the Edit long-term backup retention period dialog, and click Save. The long-term backup's Expiration date updates accordingly. • If the parent ACD has retention lock enabled, the retention period can only be increased from the original value. • If the retention lock is not enabled, you can either increase or decrease the retention period by days, months, or years.

Action	Steps
Delete a long-term backup	Click the actions icon, and select Delete. Click Delete on the confirmation dialog that appears. The backups State changes to Deleted. • If the parent ACD has retention lock enabled, then the retention lock applies to all long-term backups for the Autonomous AI Databases inside the ACD. These long-term backups cannot be deleted and can only expire. If you choose NFS as a backup destination type while creating a long-term backup, then you have to manually delete the backups after they expire if the corresponding ACD is already terminated. • If the retention lock is not enabled, the long-term backups can be deleted at any time or can be allowed to expire. You can also choose to delete them when the Autonomous AI Database gets terminated. However, if you have configured NFS for storing the long-term backup and the corresponding ACD is terminated, then you have to manually delete the backups after they expire. Note A long-term backup cannot be deleted for 7 days from the date of creation, as the Object storage bucket the backup is in has a 7-day lock.

You can see all the long-term Autonomous AI Database backups in your current compartment under **Autonomous standalone backups** on the OCI console. The long-term backups continue to be listed under **Autonomous standalone backups** even after the corresponding Autonomous AI Database is terminated.

Managing Primary and Standby Databases in an Autonomous Data Guard Configuration

Learn how to manage primary and standby databases in an autonomous data guard configuration.

When you create an Autonomous AI Database in an Autonomous Container Database that has Autonomous Data Guard enabled, two completely separate copies of your database are created: one in a *primary* container database, and one (a synchronized copy) in a *standby* container database. Then, should the primary container database become unavailable, Autonomous Data Guard automatically converts the standby container database to the primary container database and, as such, it begins servicing application connections to your Autonomous AI Database.

Note

Because two Autonomous AI Databases are created when you use Autonomous Data Guard, twice the number of CPU and storage resources are used, half for the primary database and half for the standby database.

These two databases, often called *peer* databases of each other, are identified by the labels **Primary** and **Standby** in the list of Autonomous AI Databases and on the Details page for the database.

For information about how fleet administrators create and manage Autonomous Container Databases with Autonomous Data Guard enabled, see *Manage Autonomous Data Guard*.

Management Operations on Primary and Standby Databases

Because the two databases are linked and synchronized, some of the management operations you perform on Autonomous AI Databases work differently on the primary and standby databases in an Autonomous Data Guard configuration as compared to standard databases. The following list describes these differences.

- **Scale up, Scale down, and auto-scaling**
The two peer databases must have the same CPU count, storage size, and auto-scaling setting. Therefore, you can only scale up, scale down and change auto-scaling settings from the Details page of the primary database. Changes you make affect both the primary and standby databases.
- **Stop, Start and Restart**
The lifecycle status of the two peer databases is kept synchronized. Therefore, you can only stop, start and restart the database from the Details page of the primary database. The operation you perform affects both the primary and standby databases.
- **Back up manually**
You can manually back up the primary and standby databases separately and independently, just as though they were standard databases.
- **Restore**
You can restore and recover the database only from the Details page of the primary database.
- **Clone**
You can clone the database only from the Details page of the primary database.
- **Rotate the encryption key**
You can rotate the encryption keys of the primary and standby databases separately and independently, just as though they were standard databases.
- **Move to a different compartment**
You can move the primary and standby databases to different compartments separately and independently, just as though they were standard databases.
- **Terminate**
You can terminate the database only from the Details page of the primary database.

Accessing Standby Databases from Client Applications

When using Autonomous Data Guard, your client applications normally connect to and perform operations on the primary database.

In addition to this normal connectivity, Autonomous Data Guard provides you the option to connect client applications **that perform only read-only operations** to the standby database.

To take advantage of this option, client applications connect to the database using database service names that include "_RO" (for "read only"), as described in [Predefined Database Service Names for Autonomous AI Databases](#).

Monitoring Lag Times

As your databases that use Autonomous Data Guard are running, you can monitor transport lag and apply lag times from the primary or standby database's Details page by clicking **Autonomous Data Guard**.

You should expect to see minor fluctuations over time as the workload on your database ebbs and flows. However, if you notice a continuing upward trend in lag time, you can take these actions to resolve the situation:

- **Upward Trend in Apply Lag.** A continuing upward trend in apply lag indicates that the standby database doesn't have sufficient capacity to keep up with the redo records coming from the primary database. To resolve this situation, scale up the CPUs of the database, as described in [Add CPU or Storage Resources to a Dedicated Autonomous AI Database](#).
- **Upward Trend in Transport Lag.** A continuing upward trend in transport lag indicates a network performance issue. Oracle Cloud operations staff constantly monitors network performance, so you should see the situation resolve itself without you taking any action. However, if you want, you can bring the situation to the operations staff by raising a service request.

Move an Autonomous AI Database to a Different Compartment

Shows you the steps to move an Autonomous AI Database to a different Oracle Cloud Infrastructure compartment.

Note

As soon as you move an Autonomous AI 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.

Required IAM Policies

- use autonomous-databases in the Autonomous AI Database's current compartment and in the compartment you are moving it to.
- read autonomous-backups

Procedure

1. Go to the **Details** page of the Autonomous AI Database you want to move.
For instructions, see [View Details of a Dedicated Autonomous AI Database](#).
2. On Oracle Public Cloud, click **Move resource** under **More actions**, and on Exadata Cloud@Customer, click **Move resource** under **Actions**.
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.

Restore and Recover Autonomous AI Database on Dedicated Exadata Infrastructure

You can restore and recover an Autonomous AI Database on Dedicated Exadata Infrastructure from its **Details** page.

Note

In an Autonomous Data Guard setup, database restore is not allowed if the standby database is in snapshot standby role. You must convert the standby Autonomous Container Database (ACD) to physical standby role to restore this database. See [Convert Snapshot Standby to Physical Standby](#) for instructions.

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

Required IAM Policies

use autonomous-databases

read autonomous-backups

Procedure

To restore and recover your database, do the following:

1. Go to the **Details** page of the Autonomous AI Database you want to restore and recover.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

For instructions, see [View Details of an Autonomous AI Database on Dedicated Exadata Infrastructure](#).

2. On Oracle Public Cloud, click **Restore** under **More actions**, and on Exadata Cloud@Customer, click **Restore** under **Actions**.
3. In the **Restore** prompt, choose from the following options:
 - **Enter Timestamp:** Choose this option if you want to restore a backup with a specific timestamp. After selecting this option, you must enter the timestamp in the **Enter Timestamp** field.
 - **Select Backup:** This option lets you select a backup from the list of backups. You can limit the number of backups you see by specifying a period using the **From** and **To** calendar fields. After selecting this option, you must select a backup from the list that shows up.
 - **Enter SCN:** Choose this option if you want to restore the database to a specific System Change Number (SCN). After selecting this option, you must enter the SCN in the Enter SCN field.
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. For all three types of restore (by SCN, by timestamp or by using specific backup) an SCN will be returned when the restore is completed.

Note

- When your database is restored, the value of the ADMIN user password is also restored. Therefore, it may have an old value that you no longer remember. You can set the password to a new value as described in [Unlock or Change the ADMIN Database User Password](#).
- 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.

Start, Stop, and Restart an Autonomous AI Database

You can manage start, stop, and restart an Autonomous AI Database from its **Details** page.

Required IAM Policies

use autonomous-databases

Stop an Autonomous AI Database

1. Go to the **Details** page of the Autonomous AI Database you want to stop.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

For instructions, see [View Details of a Dedicated Autonomous AI Database](#).

2. On Oracle Public Cloud, click **Stop** under **More actions**, and on Exadata Cloud@Customer, click **Stop** under **Actions**.
3. Click **Stop** to confirm.

Note

When you stop an Autonomous AI Database, 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.

Start an Autonomous AI Database

1. Go to the **Details** page of the Autonomous AI Database you want to start.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

For instructions, see View Details of a Dedicated Autonomous AI Database.

2. On Oracle Public Cloud, click **Start** under **More actions**, and on Exadata Cloud@Customer, click **Start** under **Actions**.

Start is only shown for a stopped database.

3. Click **Start** to confirm.

Note

When you start an Autonomous AI Database, CPU billing is initiated, billed by the second with a minimum usage period of one minute.

Restart an Autonomous AI Database

1. Go to the **Details** page of the Autonomous AI Database you want to restart.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

For instructions, see View Details of a Dedicated Autonomous AI Database.

2. On Oracle Public Cloud, click **Restart** under **More actions**, and on Exadata Cloud@Customer, click **Restart** under **Actions**.

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

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

Terminate an Autonomous AI Database on Dedicated Exadata Infrastructure

Shows you the steps to terminate an Autonomous AI Database on Dedicated Exadata Infrastructure.

Required IAM Policies

manage autonomous-databases

Procedure

1. Go to the **Details** page of the Autonomous AI Database you want to terminate.

✓ Tip

For databases that use Autonomous Data Guard, go to the **Details** page of the *primary* database.

ⓘ Note

In an Autonomous Data Guard setup, you can not terminate an Autonomous AI Database in an Autonomous Container Database (ACD) whose standby ACD is in a snapshot standby role.

For instructions, see View Details of a Dedicated Autonomous AI Database.

2. On Oracle Public Cloud, click **Terminate** under **More Actions**. On Exadata Cloud@Customer, click **Delete** under **Actions**.
3. On the **Terminate Autonomous AI Database** page enter the database name to confirm that you want to terminate the database.
4. Optionally, select **Delete all long-term backups that are not retention locked** to remove all long-term backups of the Autonomous AI Database if they are not retention locked.
5. Click **Terminate** in Oracle Public Cloud and **Terminate Autonomous AI Database** in Exadata Cloud@Customer.
6. After the Autonomous AI Database is terminated, you are given an option to undelete the database if the Autonomous AI Database is created with an ECPU compute model.

Undelete an Autonomous AI Database

You can only undelete an Autonomous AI Database created with an ECPU compute model.

- You can undelete an Autonomous AI Database within 72 hours of its termination, provided the corresponding ACD is not terminated.
- Undeleting a database will bring the database back to the latest available timestamp.
- Click **undelete database**, and when prompted, confirm to undelete the database.

Note

After you undelete an Autonomous AI Database, the backups created before you undeleted the database cannot be used for a database restore and can only be used for cloning the database. If you had chosen to remove all long-term backups while deleting the Autonomous AI Database (by selecting **Delete all long-term backups that are not retention locked**), then those long-term backups will not be restored.

View Autonomous AI Database Metrics for a Database

Shows the steps to view the Autonomous AI Database metrics.

Required IAM Policies

- read metrics
- inspect work-requests

See Prerequisites for Monitoring Autonomous AI Database Metrics for more details.

View Autonomous AI Database Metrics for a Database

1. Go to the **Details** page of the Autonomous AI Database whose metrics you want to view.
For instructions, see View Details of a Dedicated Autonomous AI Database.
2. Click **Monitoring** to see the metrics information. There is a chart for each metric. In each chart you can select the **Interval** and **Statistic** or use the default values.

To create an alarm on a metric, click **Options** and select **Create an Alarm on this Query**. See [Managing Alarms](#) for information on setting and using alarms.

View Autonomous AI Database Metrics for a Database

Shows the steps to view metrics for Autonomous AI Database in a compartment.

Required IAM Policies

- read metrics
- inspect work-requests

See Prerequisites for Monitoring Autonomous AI Database Metrics for more details.

View Autonomous AI Database Metrics for Databases in a Compartment

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, then click **Monitoring** and then click **Service Metrics**.
3. If needed, switch to the region hosting the Autonomous AI Databases whose metrics you want to view.

See [Switching Regions](#) for information on switching regions and working in multiple regions.

4. If needed, switch to the compartment hosting the Autonomous AI Databases whose metrics you want to view.
See [Understanding Compartments](#) for information on using and managing compartments.
5. Under **Metric Namespace** select **oci_autonomous_database**.
6. Specify **Dedicated** as the **deploymentType** dimension.
 - a. Next to **Dimensions** select **Add**.
 - b. In the **Dimension Name** field select **deploymentType**.
 - c. In the **Dimension Value** field select **Dedicated**.
 - d. Click **Done**.
7. If there are multiple Autonomous AI Databases in the compartment you can show metrics aggregated across the Autonomous AI Databases by selecting **Aggregate Metric Streams**.
8. If you want to limit the metrics you see, add more dimensions.

To create an alarm on a specific metric, click **Options** and select **Create an Alarm on this Query**. See [Managing Alarms](#) for information on setting and using alarms.

Connect to Autonomous AI Database

Download Client Credentials

Oracle client credentials (wallet files) are downloaded from Autonomous AI Database by a service administrator. If you are not an Autonomous AI Database 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 AI Database details page.
2. On Oracle Public Cloud, click **Database connection**, and on Exadata Cloud@Customer, click **DB Connection** under **Actions**.
3. On the corresponding connection page, click **Download Wallet**.
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.

Note

For cross-region standby Autonomous AI Databases, you can download region-specific connection strings by selecting **Download region-specific connection strings only** in the **Download Wallet** dialog.

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

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

You must protect this file to prevent unauthorized database access.

The zip file includes the following:

- `tnsnames.ora` and `sqlnet.ora`: Network configuration files storing connect descriptors and SQL*Net client-side configuration.
- `cwallet.sso` and `ewallet.p12`: Auto-open SSO wallet and PKCS12 file. PKCS12 file is protected by the wallet password provided on the OCI console.
- `truststore.jks`: Java truststore file that is protected by the wallet password provided while downloading the wallet.
- `ojdbc.properties`: Contains the wallet related connection property required for JDBC connection. This should be in the same path as `tnsnames.ora`.
- `hostinfo.json`: Host information file with a list of IP addresses that are part of the cluster used by the Autonomous AI Database.

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 AI 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. See Download Client Credentials for information on downloading client credentials for Autonomous AI Database.

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.

Download, Install and Configure Oracle Instant Client

Follow these steps to download and install Oracle Instant Client and then download the client credentials for your Autonomous AI Database and make 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
```
 - **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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as follows. If you don't have such an account, you need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

 - a. In your web browser, sign in to Oracle Cloud and navigate to the Details page for the Autonomous AI 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 TLS Walletless Connections

You can connect your database applications or tools to an Autonomous AI Database on Dedicated Exadata Infrastructure without a wallet. Connecting an application without a wallet

(TLS) provides security for authentication and encryption, and security is enforced using a security certificate that is trusted by the client operating system (OS).

TCPS connection without using a client wallet will only work when the following requirements are met:

1. *One-way TLS connections are enabled.*

By default, one-way TLS connections are enabled when you provision an AVMC. See [Create an Autonomous Exadata VM Cluster](#) for more information.

2. *Server SSL certificate is trusted by the client operating system.*

Use a (BYOC) digital SSL certificate signed by a well-known public CA so that it is trusted by the client OS by default. If the digital certificate is not signed by a well-known public CA such as Digicert, manually add the certificate so the client OS trusts it.

For example, in a Linux environment, add the certificate presented by the server to the `/etc/ssl/certs/ca-bundle.crt` file.

To bring your own certificate (BYOC), follow the steps described below:

- Get an SSL certificate from a public CA, such as Digicert. See [Additional Information](#) for detailed instructions.
 - Seed the SSL certificate using the OCI Certificate Service. Refer to [Creating a Certificate](#).
- These certificates must be signed and must be in the PEM format, that is, their file extension must be `.pem`, `.cer`, or `.crt` only.
- Add the SSL certificate to your AVMC from the **Manage Certificates** dialog accessible from the AVMC **Details** page. Refer to [Manage Security Certificates for an Autonomous Exadata VM Cluster](#).

Additional Information

The high-level steps involved in getting an SSL certificate from a public CA are:

1. Create a wallet.

```
WALLET_PWD=<password>
```

```
CERT_DN=CERT_DN="CN=adb.example.oraclecloud.com,OU=Oracle BMCS
FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US"
CERT_VALIDITY=365
KEY_SIZE=2048
SIGN_ALG="sha256"
WALLET_DIR=$PWD
ASYM_ALG="RSA"
```

```
$ORACLE_HOME/bin/orapki wallet create -wallet $WALLET_DIR -pwd $WALLET_PWD
-auto_login
```

2. Create a signing request (this creates a private key inside the wallet and a requested certificate)

```
$ORACLE_HOME/bin/orapki wallet add -wallet $WALLET_DIR -dn "$CERT_DN" -
pwd $WALLET_PWD
-keysize $KEY_SIZE -sign_alg $SIGN_ALG -validity $CERT_VALIDITY -
asym_alg $ASYM_ALG
```

3. Export the signing request

```
$ORACLE_HOME/bin/orapki wallet export -wallet $WALLET_DIR -dn "$CERT_DN" -
request
$WALLET_DIR/cert.csr
```

4. Send the signing request file *cert.csr* to the public CA for the CA to validate it and send back the user/leaf certificate and the chain.

5. Add the user certificate and the chain (root + intermediate certificates) in the wallet

```
$ORACLE_HOME/bin/orapki wallet add -wallet $WALLET_DIR -pwd $WALLET_PWD -
trusted_cert -cert
$WALLET_DIR/root.crt$ORACLE_HOME/bin/orapki wallet add -
wallet $WALLET_DIR -pwd $WALLET_PWD -trusted_cert -cert
$WALLET_DIR/intermediate.crt$ORACLE_HOME/bin/orapki wallet add -
wallet $WALLET_DIR -pwd $WALLET_PWD -user_cert -cert
$WALLET_DIR/usercert.crt
```

6. Upload the user certificate, chain certificates and the private key to the Oracle Cloud Infrastructure (OCI) Certificate Service. You can get the private key from the wallet using the following command:

```
openssl pkcs12 -in $WALLET_DIR/ewallet.p12 -out $WALLET_DIR/private.pem -
nocerts
```

Connect Python Applications Using mTLS

APPLIES TO:  Exadata Cloud@Customer only

You can connect Python applications to your Autonomous AI Database instance using mTLS.

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

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

Follow these steps to connect your Python application to an Autonomous AI Database instance using mTLS:

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

Install Python and the python-oracledb Driver

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

1. Install [Python 3](#), if it is not already available.

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

Note

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

2. Install the python-oracledb driver from [PyPI](#).

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

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

Run the following command to upgrade python:

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

You should see output similar to the following:

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

Notes for installing python-oracledb:

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

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

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

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

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

See [Installing python-oracledb](#) for additional options and tips.

3. If you want to use the python-oracledb driver in Thick mode, install Oracle Client software. By default, python-oracledb runs in Thin mode which connects directly to Oracle Database. Thin mode does not require Oracle Client libraries. However, some additional functionality is available when python-oracledb runs in Thick mode.

Note

See [Oracle Database Features Supported by python-oracledb](#) for information on supported features in python-oracledb Thin and Thick modes. Not all of the features shown in this link are available with Autonomous AI Database.

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

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

- **Mutual TLS (mTLS) Connections:**
 - If your database is on a remote computer, then download the free [Oracle Instant Client](#) “Basic” or “Basic Light” package for your operating system architecture. Use a supported version: **Oracle Instant Client:** 18.19 (or later), 19.2 (or later), or 21 (base release or later).
 - Alternatively, you can use the Full Oracle Database client libraries when they are available on your system (including Full Oracle Database Client: **Oracle Database Client:** 18.19 (or later), 19.2 (or later), or 21 (base release or later).
- **TLS Connections:** Oracle Call Interface (OCI) clients support TLS authentication if you are using the following client versions:
 - Oracle Instant Client/Oracle Database Client 19.14 (or later) and 21.5 (or later) - all platforms
 - Alternatively, you can use the Full Oracle Database client libraries when they are available on your system, including Full Oracle Database Client 19.14 (or later) and 21.5 (or later).

Obtain Security Credentials (Oracle Wallet) and Enable Network Connectivity

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

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

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

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

Note

Protect the `wallet.zip` file and its contents to prevent unauthorized database access.

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

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

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

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

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

To connect in Thin mode:

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

-
- [Linux](#)
 - [Windows](#)

Linux

For example on Linux:

`/opt/OracleCloud/MYDB`

Windows

For example on Windows:

`C:\opt\OracleCloud\MYDB`

-
2. In your Python application, set the following connection parameters to connect to an Autonomous AI Database instance:
 - `config_dir`: Specifies the directory containing `tnsnames.ora`.
 - `dsn`: Use to specify the desired network alias from the `tnsnames.ora` file.
 - `password`: Specifies the database user password.

- `user`: Specifies the database user.
- `wallet_location`: Specifies the directory containing the PEM file (`ewallet.pem`).
- `wallet_password`: Specifies the password for the PEM file (`ewallet.pem`). You set this password when you download the `wallet.zip` file.

-
- [Linux](#)
 - [Windows](#)

Linux

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

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

Windows

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

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

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

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

If you are behind a firewall, you can tunnel TLS/SSL connections through a proxy using `HTTPS_PROXY` in the connect descriptor or by setting connection attributes. Successful connection depends on specific proxy configurations. Oracle does not recommend using a proxy in a production environment, due to the possible impact on performance. See `HTTPS_PROXY` in Oracle Database 19c Database Net Services Reference or [Oracle Database 26ai Database Net Services Reference](#) for more information.

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

For example, on Linux:

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

For example, on Windows:

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

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

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

Note

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

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

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

To connect in Thick mode:

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

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

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

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

or

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

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

- Alternatively, move the files to any accessible directory.

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

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

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

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

Note

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

2. In your Python application set the following initialization and connection parameters to connect to the Autonomous AI Database instance:
 - config_dir: Specifies the configuration directory when you are putting the configuration files. This is only required when the configuration files are placed in a directory outside of the instant client configuration directory network/admin.
 - dsn: Specifies the desired network alias from the tnsnames.ora file.
 - password: Specifies the database user password.
 - user: Specifies the database user.

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

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

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

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

-
- [Linux](#)
 - [Windows](#)

Linux

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

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

Windows

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

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

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

If you are behind a firewall, you can tunnel TLS/SSL connections through a proxy using `HTTPS_PROXY` in the connect descriptor or by setting connection attributes. Successful connection depends on specific proxy configurations. Oracle does not recommend using a proxy in a production environment, due to the possible impact on performance. See [HTTPS_PROXY](#) in Oracle Database 19c Database Net Services Reference or [Oracle Database 26ai Database Net Services Reference](#) for more information.

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

```
SQLNET.USE_HTTPS_PROXY=on
```

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

For example:

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

See [Enabling python-oracledb Thick mode](#) for information on Thick mode.

Connect with Node.js and other Scripting Languages

You can use programs in different languages, including Node.js, PHP, Ruby, R, Go, and Perl to connect to Autonomous AI Database on Dedicated Exadata Infrastructure. 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 Autonomous AI Database on Dedicated Exadata Infrastructure from your scripting language, first install the language driver and client libraries as follows:

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

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

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

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

- b. If you are building a language API or driver from source code, you may also need to download the Instant Client SDK: [Oracle Instant Client](#)
- c. Unzip the base package you selected. For example unzip to C:\instantclient_12_2 or /home/myuser/instantclient_18_5. If you also download the SDK, unzip it in the same directory.
- d. On Windows, add the path to the PATH variable in the "System variables" section of the Environment Variables pane (for example add C:\instantclient_12_2). On Windows 8 access the PATH variable setting area by navigating to Control Panel>System>Advanced System Settings>Environment Variables. If you have multiple versions of Oracle libraries installed make sure the new directory occurs first in the path.
- e. On non-Windows platforms, create a symbolic link if it does not exist. For example:

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

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

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

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

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

Note

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

2. Install the relevant language driver for Oracle Database:

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

Windows DLLs are available on <http://php.net/downloads.php> and are also available from [PECL oci8](#).

- **PHP PDO_OCI**: To install PHP PDO_OCI for PHP, use the instructions on the following page: [Oracle Functions \(PDO_OCI\)](#).

Windows DLLs are available on <http://php.net/downloads.php> included in PHP.

- **Ruby**: To install ruby-oci8 for Ruby, use the instructions on the following page: [Install for Oracle Instant Client](#)
- **DBD for Perl**: To install DBD::Oracle for Perl, set ORACLE_HOME and your library search path such as LD_LIBRARY_PATH or PATH to the Instant Client directory and use the instructions on the following page: [DBD::Oracle](#).

Enable Oracle Network Connectivity and Obtain the Security Credentials (Oracle Wallet)**1. Obtain client security credentials to connect to an Autonomous AI Database instance. You obtain a zip file containing client security credentials and network configuration settings required to access your Autonomous AI 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 AI Database.

2. Extract the client credentials (wallet) files:

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

C:\instantclient_12_2\network\admin

or

/home/myuser/instantclient_18_5/network/admin

or

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

- If you are using a full Oracle Client move the file to \$ORACLE_HOME/network/admin.
- c. Alternatively, put the unzipped wallet files in a secure directory and set the TNS_ADMIN environment variable to that directory name.

Note

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

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 Database Actions

Tip

For a "try it out" alternative that demonstrates these instructions, run **Lab 7: Database Actions Console** in the Oracle Autonomous AI Database Dedicated Workshop.

Oracle Database Actions, a browser-based application that provides many database development, management and monitoring features is built into Autonomous AI Databases. Therefore, you can use it without downloading or installing additional software on your system. Database Actions was earlier known as SQL Developer Web.

Requirements

To be able to sign into Database Actions:

- The system you are using must have network access to the Autonomous AI Database. This can be achieved using either of the following ways:
 - By Configuring a development system to access the database.
 - By setting proper network routes with VPN (Fast connect or VPN as a Service) or any VCN peering if the compute host is in another VCN.
- You must have an Oracle Cloud user account that permits you to access the database's **Details** page in the Oracle Cloud console page.
- If you are an application developer, the ADMIN user must enable your access to Database Actions, as described in Enabling User Access to Database Actions in *Using Oracle*

Database Actions. This is required because only the ADMIN database user has access to Database Actions, by default.

Access Database Actions

To access the Database Actions **Sign In** page, you copy its URL from the **Database Actions** dialog found in the **Tools** tab of the database's **Details** page, and then paste the URL into your web browser's address bar.

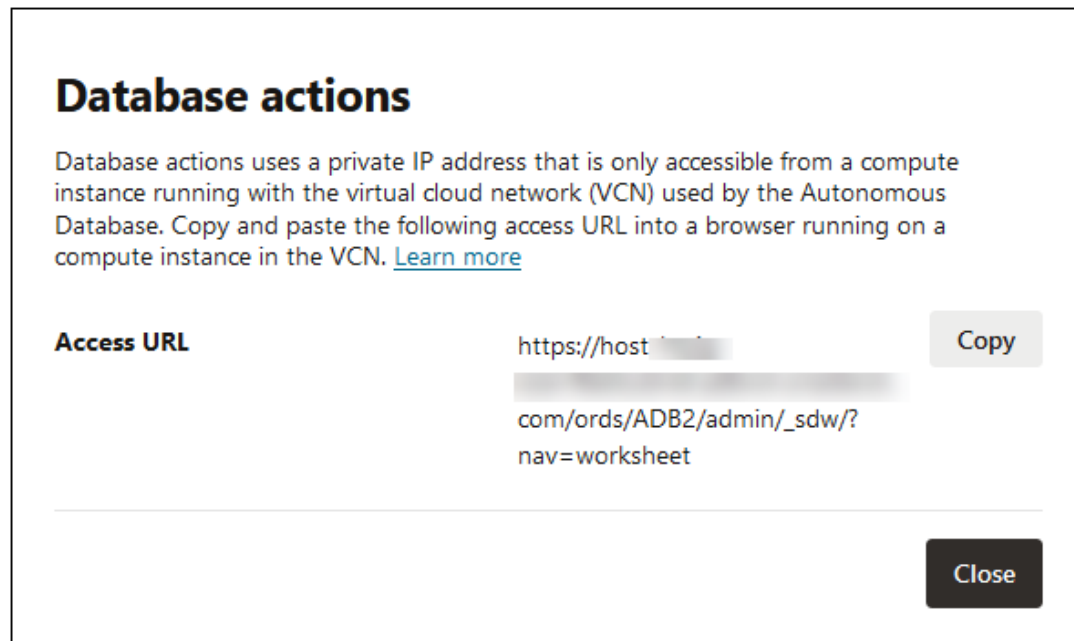
1. In your web browser, sign in to Oracle Cloud and navigate to the **Details** page for the Autonomous AI Database.

For instructions, see [View Details of a Dedicated Autonomous AI Database](#).

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the primary database. Database Actions can not be launched from a standby database in an Autonomous Data Guard configuration.

2. Click **Database Actions**.
3. In the **Database Actions** dialog, click the **Copy** link in the **Access URL** field to copy the URL to your clipboard.



4. If you are an application developer, paste the URL into a text editor and then change "admin" to the user name of the database user you are signing in as. Copy the edited URL and paste it into your web browser's address bar.

However, as a database administrator, you can paste the copied URL into your web browser's address bar directly.

Connect with SQL*Plus

✓ Tip

For a "try it out" alternative that demonstrates these instructions, run **Task 4: Connect to your autonomous DB using SQL Worksheets from Database Actions, SQLCL and SQL Plus** from *Lab 2: Configure a Development System* in the Oracle Autonomous AI Database Dedicated Workshop.

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

To be able to connect SQL*Plus to an Autonomous AI Database, the system running SQL*Plus must have network access to the Autonomous AI Database. This can be achieved using either of the following ways:

- By Configuring a development system to access the database.
- By setting proper network routes with VPN (Fast connect or VPN as a Service) or any VCN peering if the compute host is in another VCN.

To install and configure the client and connect to an Autonomous AI Database using SQL*Plus, do the following:

1. Prepare to connect by following the instructions in Prepare for ODBC and JDBC Oracle Call Interface (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
```

```
SQL*Plus: Release 18.0.0.0.0 - Production on Mon Jan 18 09:50:13 2021  
Version 18.5.0.0.0
```

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

```
Enter password:
```

```
Last Successful login time: Mon Jan 18 2021 09:33:57 +00:00
```

```
Connected to:
```

```
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production  
Version 19.0.0.0.0
```

```
SQL>
```

📘 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.

Connect with Oracle SQLcl

✓ Tip

For a "try it out" alternative that demonstrates these instructions, run **Task 4: Connect to your autonomous DB using SQL Worksheets from Database Actions, SQLCL and SQL Plus** from *Lab 2: Configure a Development System* in the Oracle Autonomous AI Database Dedicated Workshop.

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.

To be able to connect SQLcl to an Autonomous AI Database, the system running SQLcl must have network access to the Autonomous AI Database. This can be achieved using either of the following ways:

- By Configuring a development system to access the database.
- By setting proper network routes with VPN (Fast connect or VPN as a Service) or any VCN peering if the compute host is in another VCN.

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

SQLcl can connect to an Autonomous AI 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 ODBC and JDBC Oracle Call Interface (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 19.1 Production on Mon Jan 18 09:28:38 2021
```

```
Copyright (c) 1982, 2021, Oracle. All rights reserved.
```

```
Username? ('') sales_trans@atpc1_low
```

```
Password? (*****?) *****
```

```
Last Successful login time: Mon Jan 17 2019 15:29:19 -07:00
```

```
Connected to:
```

```
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
```

```
Version 19.9.0.0.0
```

```
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 AI 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 AI Database:

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

For example:

```
sql /nolog
```

```
SQLcl: Release 19.1 Production on Mon Jan 18 09:28:38 2021
```

```
Copyright (c) 1982, 2021, Oracle. All rights reserved.
```

```
SQL> set cloudconfig /home/atpc/wallet_ATPC.zip  
Operation is successfully completed.  
Operation is successfully completed.  
Using temp directory:/tmp/oracle_cloud_config3643858537577169472
```

```
SQL> connect admin@atpc_medium  
Password? (*****?) *****  
Connected.
```

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

For information on SQLcl, see *Oracle SQLcl*.

Connect to Autonomous AI Database on Dedicated Exadata Infrastructure with Oracle SQL Developer

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

Requirements

To be able to connect SQL Developer to an Autonomous AI Database, the system running SQL Developer must have network access to the Autonomous AI Database. This can be achieved using either of the following ways:

- By Configuring a development system to access the database.
- By setting proper network routes with VPN (Fast connect or VPN as a Service) or any VCN peering if the compute host is in another VCN.

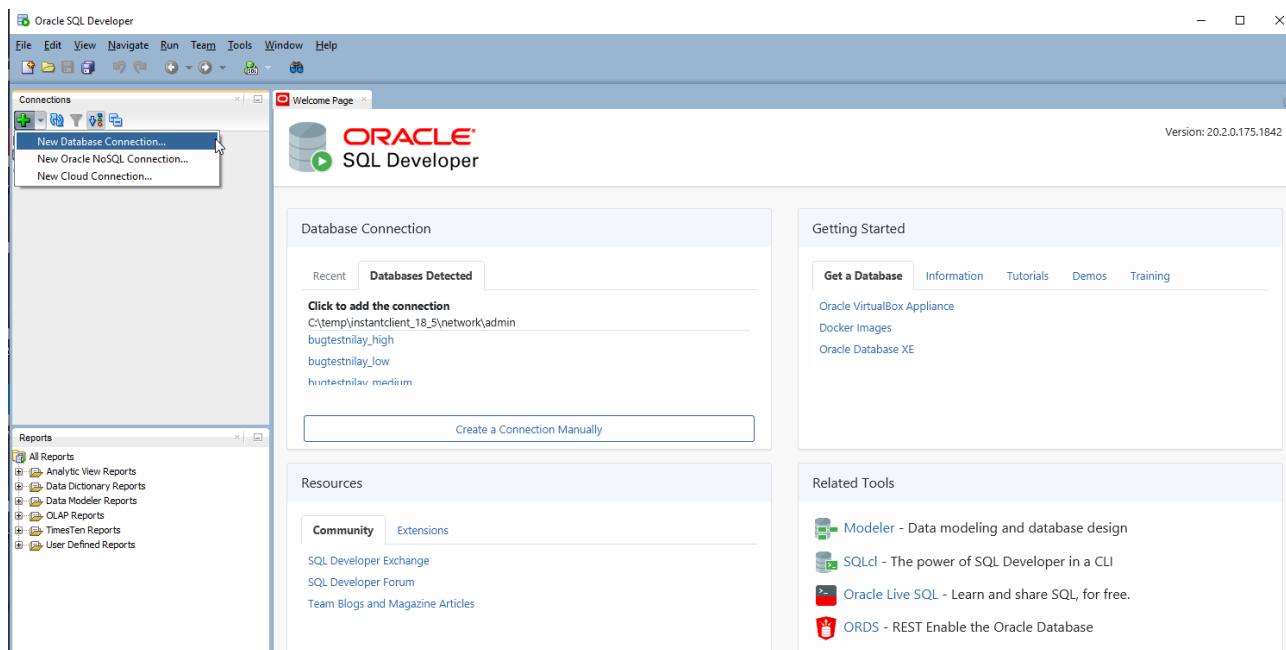
Connect with Oracle SQL Developer Version 18.2 or Later

To create a new connection to an Autonomous AI 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. If you have an Oracle Cloud user account that permits you to access this page, download the credentials as described in [Download Client Credentials](#). Otherwise, you will need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

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 recommends that you use Oracle SQL Developer version 18.2 or later; however, earlier versions of SQL Developer will work with Autonomous AI Database.

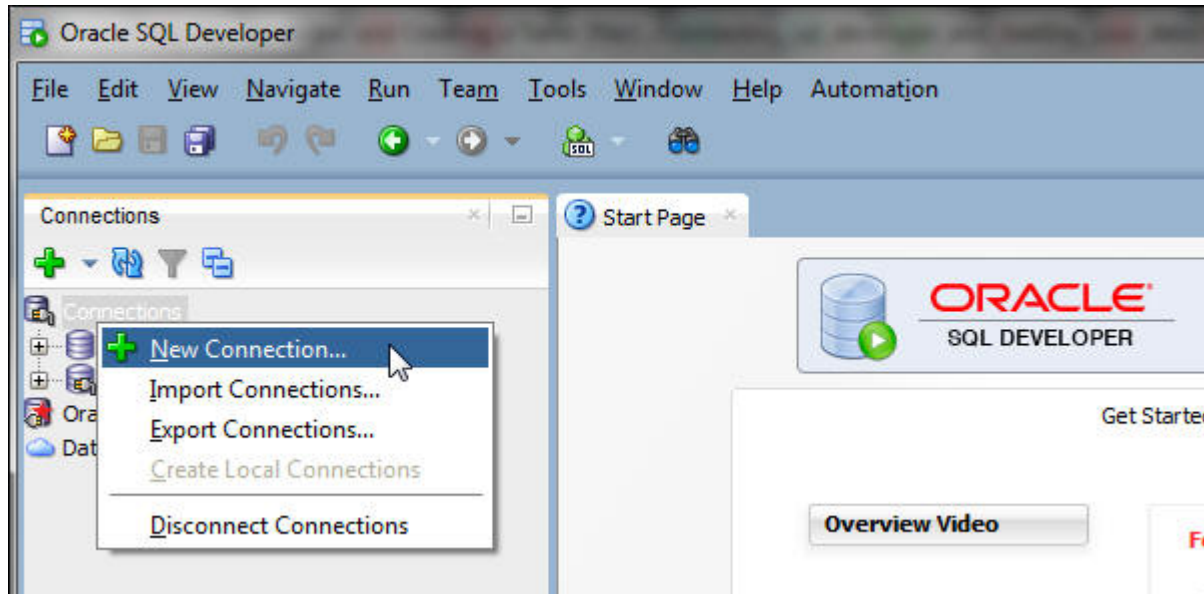
To create a new connection to an Autonomous AI Database, do the following:

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

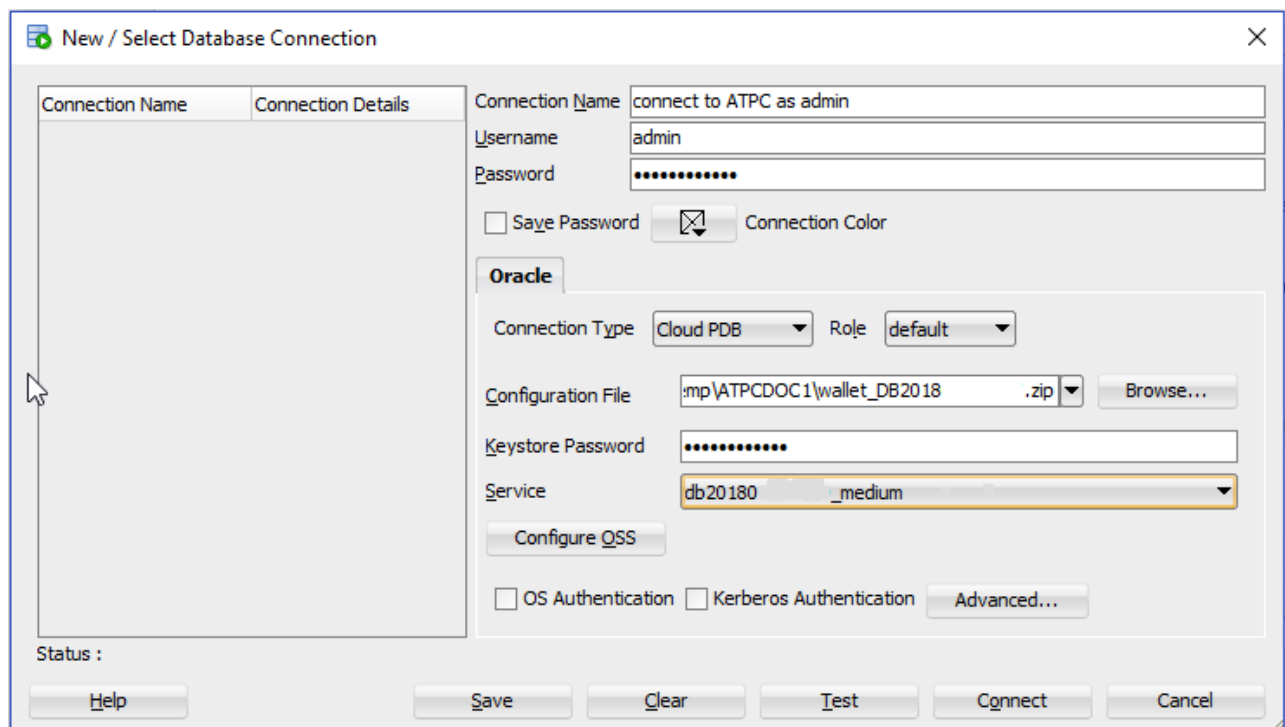
This zip file is available for download from the database's Details page in the Oracle Cloud console. If you have an Oracle Cloud user account that permits you to access this page,

download the credentials as described in Download Client Credentials. Otherwise, you will need to get the zip file from the administrator of the database, together with the password that was used to protect the zip file.

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 AI Database.
- **Service:** Enter the service name. The client credentials file provides the service names.

View Connection Strings for an Autonomous AI Database

You can view the TNS names and connection strings of an Autonomous AI Database from its **Details** page on the Oracle Cloud Infrastructure (OCI) console.

Required IAM Policies

`inspect autonomous-databases`

Procedure

To view the connection strings for an Autonomous AI Database:

1. Go to the **Details** page of the an Autonomous AI Database for which you want to view connection strings.

For instructions, see View Details of a Dedicated Autonomous AI Database.

Note

For databases that use Autonomous Data Guard, go to the **Details** page of the primary database.

2. On Oracle Public Cloud, click **Database connection**, and on Exadata Cloud@Customer, click **DB Connection** under **Actions**.
3. The **Database connection** dialog displays connection strings in a table format with TNS name and its connection string in each row.

mTLS and TLS connections use the same TNS names, but their connection strings differ according to their port definitions. The type of connection strings shown will depend on the settings chosen when creating the parent Autonomous Exadata VM Cluster. See Create an Autonomous Exadata VM Cluster for more information.
4. Click the  menu in the row of the TNS Name whose Connection string you want to copy. Click **Copy** to copy the entire connection string.
5. For cross-region standby Autonomous AI Databases, you can select **Show region-specific connection strings** to view connection strings specific to the selected region.

Note

This option is available only when the standby database is in a region different from the primary database.

See Predefined Database Service Names for Autonomous AI Database for information about the different database connection services for which connection strings are created.

Connect to Oracle Autonomous AI Database Using JDBC with Eclipse

You can establish a connection to Oracle Autonomous AI Database using the Eclipse IDE. The following sections provide information about how to first connect to Oracle Autonomous AI Database using a Maven project or a Gradle project. Make sure that you have completed the steps described in Prerequisites prior to connecting to Oracle Autonomous AI Database.

Prerequisites

The following sections provide information about the tasks that you need to perform before connecting your Java applications to Oracle Autonomous AI Database using Oracle JDBC driver and Universal Connection Pool. Refer to the appropriate prerequisites depending on the authentication type selected during provisioning of your Autonomous Exadata VM Cluster (AVMC). By default, one-way TLS connections are enabled when you provision an AVMC. See [Create an Autonomous Exadata VM Cluster](#) for more information.

You can find the authentication type on the Details page of your AVMC. See [View Details of an Autonomous Exadata VM Cluster](#) for instructions.

- [One-way TLS](#)
- [mTLS](#)

One-way TLS

Provision an Oracle Autonomous AI Database Instance

You need access to an Oracle Autonomous AI Database. Refer to [Before You Begin with Autonomous AI Database on Dedicated Exadata Infrastructure](#) if you have no provisioned one already.

Remember the password that you used for ADMIN user. For the demonstration purpose, we will be using ADMIN user, but our recommendation is to create other database users either using Oracle SQL Developer or Database Actions.

Install JDK 8

Download latest [JDK 8](#) or a [higher JDK versions](#).

Note

Ensure that you use JDK8u162 or a later version. Use `java -version` to check the JDK version that you have installed. To check the JDBC driver version, type `java -jar ojdbc8.jar`.

Download a Sample Program from Github

1. Download the [ADBQuickStart.java](#) file from Github. This sample application uses the Sales History (SH) sample schema and displays 20 records from the SH.CUSTOMERS table.
2. Modify the ADBQuickStart.java file to include your Oracle Autonomous AI Database connection information:
 - **DB_USER**: You can use ADMIN, the user created by default when the Oracle Autonomous AI Database is created (if you create another Oracle Autonomous AI Database user, you can use that user instead).
 - **DB_PASSWORD**: Use the database user's password. If connecting as the ADMIN user, set this to the password you chose during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. For security reasons, you need to enter the password through the console when you run the sample.
 - **DB_URL**: Obtain the connection string for your Autonomous AI Database by following the instructions in View Connection Strings for an Autonomous AI Database. If you are directly using in the Java program, you need to escape " in the connection string with \"

A sample quick start Java file looks like this:

```
DB_URL = "jdbc:oracle:thin:@jdbc:oracle:thin:@(description=
    (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1521)
    (host=adb.us-sanjose-1.oraclecloud.com))
    (connect_data=(service_name=g13ec47eade81f7_demodb_low.adb.oraclecloud.com)
    )(security=(ssl_server_cert_dn=\"CN=adb.us-sanjose-1.oraclecloud.com,
    OU=Oracle ADB SANJOSE, O=Oracle Corporation, L=Redwood City,
    ST=California,
    C=US\")))\"
DB_USER=\"ADMIN\" and DB_PASSWORD=\"your_password\"
```

3. Save changes to the ADBQuickStart.java file.

Troubleshooting and Debugging: If you encounter any error, refer the [Troubleshooting Tips](#) page for some helpful hints.

mTLS

Provision an Oracle Autonomous AI Database Instance

You need access to an Oracle Autonomous AI Database. Refer to Before You Begin with Autonomous AI Database on Dedicated Exadata Infrastructure if you have no provisioned one already.

Remember the password that you used for ADMIN user. For the demonstration purpose, we will be using ADMIN user, but our recommendation is to create other database users either using Oracle SQL Developer or Database Actions.

Obtain Client Credentials

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

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

- **ADMIN user**: On the Oracle Cloud Infrastructure Console, click **Database connection**. See Download Client Credentials.

- Other user (non-administrator): Obtain the Oracle Wallet from the administrator for your Autonomous AI Database instance.

Note

Protect the wallet.zip file and its contents to prevent unauthorized database access.

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

Install JDK 8

Download latest [JDK 8](#) or a [higher JDK versions](#).

Note

Ensure that you use JDK8u162 or a later version. Use `java -version` to check the JDK version that you have installed. To check the JDBC driver version, type `java -jar ojdbc8.jar`.

Download a Sample Program from Github

1. Download the [ADBQuickStart.java](#) file from Github. This sample application uses the Sales History (SH) sample schema and displays 20 records from the SH.CUSTOMERS table.
2. Modify the ADBQuickStart.java file to include your Oracle Autonomous AI Database connection information:
 - **DB_USER:** You can use ADMIN, the user created by default when the Oracle Autonomous AI Database is created (if you create another OracleAutonomous AI Database user, you can use that user instead).
 - **DB_PASSWORD:** Use the database user's password. If connecting as the ADMIN user, set this to the password you chose during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. For security reasons, you need to enter the password through the console when you run the sample.
 - **DB_URL:** Enter the net service name (TNS Alias) **DBName_medium** where **DBName** is the Oracle Autonomous AI Database Name entered during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. The available net service names can be seen in the tnsnames.ora file, which is a part of the client credentials zip file.
TNS_ADMIN should point to the location where you have unzipped the client credentials of Oracle Autonomous AI Database.

A sample quick start Java file looks like this:

```
DB_URL = "jdbc:oracle:thin:@DBName_medium?TNS_ADMIN=/Users/test/  
wallet_DBName" DB_USER="ADMIN" and DB_PASSWORD="enter_it_from_console"
```

3. Save changes to the ADBQuickStart.java file.

Procedure

You can connect to Autonomous AI Database using Eclipse and either Maven or Gradle.

- [Maven](#)
- [Gradle](#)

Maven

1. Create a Maven project.
 - a. From the **File** menu, select **New**, and then select **New Maven Project**. You can either use maven archetype or select **Create a simple project (skip archetype selection)**.
 - b. Choose GAV for your Maven project. These will appear in the pom.xml file for the project.
 - **Group Id:** com.oracle
 - **Artifact Id:** ADBquickstart
 - **Version:** Leave it as 0.0.1-SNAPSHOT
2. Create the ADBQuickStart.java file.
 - a. Right-click on src/main/java, select **New**, and then select **Class**.
 - b. Enter the following values, and then click **Finish**.
 - **Package:** com.oracle.jdbctest
 - **Name:** ADBQuickStart

Note

Make sure to copy contents of the [ADBQuickStart.java](#) file to the new file created.

3. Modify the pom.xml file with the following changes: Add Oracle JDBC Driver as a dependency.
ojdbc8-production will download Oracle JDBC driver (ojdbc8.jar) along with ucp.jar (required for using UCP as a client side connection pool), oracledb.jar, osdt_core.jar, osdt_cert.jar. These JARs are required for using Oracle Wallets while connecting to Oracle Autonomous AI Database.

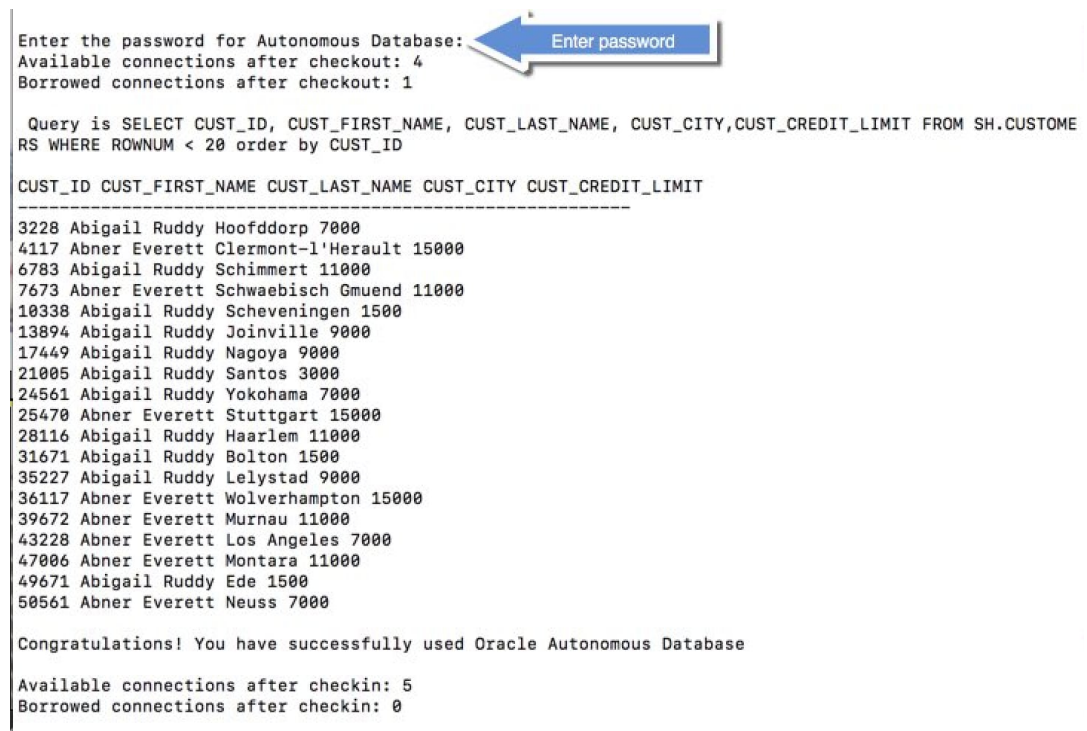
```
<properties>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
</properties>
<dependencies>
  <dependency>
    <groupId>com.oracle.database.jdbc</groupId>
    <artifactId>ojdbc8-production</artifactId>
    <version>19.18.0.0</version>
    <type>pom</type>
```

```
</dependency>
</dependencies>
```

- Build and run a sample Java program. Make sure you do not have any compilation error in the Java code and you are using the latest JDK version. Right-click **ADBQuickStart.java**. Click **Run As**, and then click **Java Application** to run the sample Java program. You are prompted to enter the database password. Once you enter the password, the results are displayed.

Sample Output:

The queried rows along with a success message are displayed, as shown in the following screen:



```
Enter the password for Autonomous Database:
Available connections after checkout: 4
Borrowed connections after checkout: 1

Query is SELECT CUST_ID, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_CITY, CUST_CREDIT_LIMIT FROM SH.CUSTOMERS WHERE ROWNUM < 20 order by CUST_ID

CUST_ID CUST_FIRST_NAME CUST_LAST_NAME CUST_CITY CUST_CREDIT_LIMIT
-----
3228 Abigail Ruddy Hoofddorp 7000
4117 Abner Everett Clermont-1'Herauld 15000
6783 Abigail Ruddy Schimmert 11000
7673 Abner Everett Schwaebisch Gmuend 11000
10338 Abigail Ruddy Scheveningen 1500
13894 Abigail Ruddy Joinville 9000
17449 Abigail Ruddy Nagoya 9000
21005 Abigail Ruddy Santos 3000
24561 Abigail Ruddy Yokohama 7000
25470 Abner Everett Stuttgart 15000
28116 Abigail Ruddy Haarlem 11000
31671 Abigail Ruddy Bolton 1500
35227 Abigail Ruddy Lelystad 9000
36117 Abner Everett Wolverhampton 15000
39672 Abner Everett Murnau 11000
43228 Abner Everett Los Angeles 7000
47006 Abner Everett Montara 11000
49671 Abigail Ruddy Ede 1500
50561 Abner Everett Neuss 7000

Congratulations! You have successfully used Oracle Autonomous Database

Available connections after checkin: 5
Borrowed connections after checkin: 0
```

Note

If you connect to Oracle Autonomous AI Database from behind a firewall, you will likely encounter a connection timeout error. Make sure to be outside the firewall while running this sample or update the `tnsnames.ora` file to use an HTTPS proxy.

Gradle

- Follow the instructions from the [Gradle Guide](#) for Gradle download and build instructions, and set the `PATH` variable before using Gradle commands. As a first step, create a Gradle project using the following command. Make sure to choose `2:application` for **Select type of project to generate**. Also, for **Source package (default:temp)**: use `com.oracle.jdbctest`.

```
gradle init
```

2. Copy the [ADBQuickStart.java](#) file to the `src/main/java/com/oracle/jdbctest` directory.
3. Modify the `build.gradle` file with the following changes:
 - Add `mavenCentral()` as a repository.
 - Add Oracle JDBC driver as a dependency.

Note

`ojdbc8-production` downloads Oracle JDBC driver (`ojdbc8.jar`) along with `ucp.jar` (required for using UCP as a client side connection pool), `oraclepki.jar`, `osdt_core.jar`, `osdt_cert.jar`. These JARs are required for using Oracle Wallets while connecting to Oracle Autonomous AI Database.

For additional information, see [Maven Central Guide](#).

- Update the `mainClassName` to `ADBQuickStart`.
- Add a run block to read the password from the console.

```
repositories {  
    // Maven Central  
    mavenCentral()  
}  
dependencies {  
    // Get the 19.18.0.0 Oracle JDBC driver along with other companion  
    jars  
    implementation("com.oracle.database.jdbc:ojdbc8-production:19.18.0.0")  
}  
application {  
    // Define the main class for the application  
    mainClassName = '{your_project_directory}.ADBQuickStart'  
}  
// To pause to read the password from console  
run {  
    standardInput = System.in  
}
```

4. Build a Gradle Application. Make sure you are in the directory where `build.gradle` file is present. Compile the Java code using the following command:

```
./gradlew build
```

5. Run the sample Java program.

```
./gradlew run
```

Sample Output:

The queried rows along with a success message are displayed, as shown in the following screen:

```

Enter the password for Autonomous Database:
Available connections after checkout: 4
Borrowed connections after checkout: 1

Query is SELECT CUST_ID, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_CITY, CUST_CREDIT_LIMIT FROM SH.CUSTOMERS
WHERE ROWNUM < 20 order by CUST_ID

CUST_ID CUST_FIRST_NAME CUST_LAST_NAME CUST_CITY CUST_CREDIT_LIMIT
-----
3228 Abigail Ruddy Hoofddorp 7000
4117 Abner Everett Clermont-1'Herault 15000
6783 Abigail Ruddy Schimmert 11000
7673 Abner Everett Schwaebisch Gmuend 11000
10338 Abigail Ruddy Scheveningen 1500
13894 Abigail Ruddy Joinville 9000
17449 Abigail Ruddy Nagoya 9000
21005 Abigail Ruddy Santos 3000
24561 Abigail Ruddy Yokohama 7000
25470 Abner Everett Stuttgart 15000
28116 Abigail Ruddy Haarlem 11000
31671 Abigail Ruddy Bolton 1500
35227 Abigail Ruddy Lelystad 9000
36117 Abner Everett Wolverhampton 15000
39672 Abner Everett Murnau 11000
43228 Abner Everett Los Angeles 7000
47006 Abner Everett Montara 11000
49671 Abigail Ruddy Ede 1500
50561 Abner Everett Neuss 7000

Congratulations! You have successfully used Oracle Autonomous Database

Available connections after checkin: 5
Borrowed connections after checkin: 0

```

Note

If you connect to Oracle Autonomous AI Database from behind a firewall, you will likely encounter a connection timeout error. Make sure to be outside the firewall while running this sample or update the `tnsnames.ora` file to use an HTTPS proxy.

Connect to Oracle Autonomous AI Database Using JDBC with IntelliJ

You can establish a connection to Oracle Autonomous AI Database using the IntelliJ IDE. The following sections provide information about how to first connect to Oracle Autonomous AI Database using a Maven project or a Gradle project. Make sure that you have completed the steps described in Prerequisites prior to connecting to Oracle Autonomous AI Database.

Prerequisites

The following sections provide information about the tasks that you need to perform before connecting your Java applications to Oracle Autonomous AI Database using Oracle JDBC driver and Universal Connection Pool. Refer to the appropriate prerequisites depending on the authentication type selected during provisioning of your Autonomous Exadata VM Cluster (AVMC). By default, one-way TLS connections are enabled when you provision an AVMC. See [Create an Autonomous Exadata VM Cluster](#) for more information.

You can find the authentication type on the Details page of your AVMC. See [View Details of an Autonomous Exadata VM Cluster](#) for instructions.

- [One-way TLS](#)
- [mTLS](#)

One-way TLS

Provision an Oracle Autonomous AI Database Instance

You need access to an Oracle Autonomous AI Database. Refer to [Before You Begin with Autonomous AI Database on Dedicated Exadata Infrastructure](#) if you have no provisioned one already.

Remember the password that you used for ADMIN user. For the demonstration purpose, we will be using ADMIN user, but our recommendation is to create other database users either using Oracle SQL Developer or Database Actions.

Install JDK 8

Download latest [JDK 8](#) or a [higher JDK versions](#).

Note

Ensure that you use JDK8u162 or a later version. Use `java -version` to check the JDK version that you have installed. To check the JDBC driver version, type `java -jar ojdbc8.jar`.

Download a Sample Program from Github

1. Download the [ADBQuickStart.java](#) file from Github. This sample application uses the Sales History (SH) sample schema and displays 20 records from the SH.CUSTOMERS table.
2. Modify the ADBQuickStart.java file to include your Oracle Autonomous AI Database connection information:
 - **DB_USER:** You can use ADMIN, the user created by default when the Oracle Autonomous AI Database is created (if you create another OracleAutonomous AI Database user, you can use that user instead).
 - **DB_PASSWORD:** Use the database user's password. If connecting as the ADMIN user, set this to the password you chose during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. For security reasons, you need to enter the password through the console when you run the sample.
 - **DB_URL:** Obtain the connection string for your Autonomous AI Database by following the instructions in View Connection Strings for an Autonomous AI Database. If you are directly using in the Java program, you need to escape " in the connection string with \"

A sample quick start Java file looks like this:

```
DB_URL = "jdbc:oracle:thin:@jdbc:oracle:thin:@(description=
    (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1521)
    (host=adb.us-sanjose-1.oraclecloud.com))
    (connect_data=(service_name=g13ec47eade81f7_demodb_low.adb.oraclecloud.com))
    )(security=(ssl_server_cert_dn=\"CN=adb.us-sanjose-1.oraclecloud.com,
    OU=Oracle ADB SANJOSE, O=Oracle Corporation, L=Redwood City,
    ST=California,
    C=US\")))\"
DB_USER=\"ADMIN\" and DB_PASSWORD=\"your_password\"
```


3. Save changes to the `ADBQuickStart.java` file.

Troubleshooting and Debugging: If you encounter any error, refer the [Troubleshooting Tips](#) page for some helpful hints.

mTLS

Provision an Oracle Autonomous AI Database Instance

You need access to an Oracle Autonomous AI Database. Refer to [Before You Begin with Autonomous AI Database on Dedicated Exadata Infrastructure](#) if you have no provisioned one already.

Remember the password that you used for ADMIN user. For the demonstration purpose, we will be using ADMIN user, but our recommendation is to create other database users either using Oracle SQL Developer or Database Actions.

Obtain Client Credentials

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

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

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

Note

Protect the `wallet.zip` file and its contents to prevent unauthorized database access.

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

Install JDK 8

Download latest [JDK 8](#) or a [higher JDK versions](#).

Note

Ensure that you use JDK8u162 or a later version. Use `java -version` to check the JDK version that you have installed. To check the JDBC driver version, type `java -jar ojdbc8.jar`.

Download a Sample Program from Github

1. Download the [ADBQuickStart.java](#) file from Github. This sample application uses the Sales History (SH) sample schema and displays 20 records from the SH.CUSTOMERS table.
2. Modify the `ADBQuickStart.java` file to include your Oracle Autonomous AI Database connection information:
 - **DB_USER:** You can use ADMIN, the user created by default when the Oracle Autonomous AI Database is created (if you create another OracleAutonomous AI Database user, you can use that user instead).

- **DB_PASSWORD:** Use the database user's password. If connecting as the ADMIN user, set this to the password you chose during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. For security reasons, you need to enter the password through the console when you run the sample.
- **DB_URL:** Enter the net service name (TNS Alias) **DBName_medium** where **DBName** is the Oracle Autonomous AI Database Name entered during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. The available net service names can be seen in the `tnsnames.ora` file, which is a part of the client credentials zip file.
TNS_ADMIN should point to the location where you have unzipped the client credentials of Oracle Autonomous AI Database.

A sample quick start Java file looks like this:

```
DB_URL = "jdbc:oracle:thin:@DBName_medium?TNS_ADMIN=/Users/test/wallet_DBName" DB_USER="ADMIN" and DB_PASSWORD="enter_it_from_console"
```

3. Save changes to the `ADBQuickStart.java` file.

Procedure

You can connect to Autonomous AI Database using IntelliJ and either Maven or Gradle.

-
- [Maven](#)
 - [Gradle](#)

Maven

1. Create a Maven project.
 - a. Click from **File** menu, select **New**, and then select **Project**.
 - b. Select **Maven** on the left hand side and select the latest version of JDK as **Project SDK**.
 - c. Click **Next**.
 - d. Give **Name** as `ADBQuickstart`.
2. Create the `ADBQuickStart.java` file.
 - a. Right-click on `src/main/java`.
 - b. Select **New**, and then select **Java Class**
 - c. Enter `com.oracle.jdbctest.ADBQuickStart.java`. This will create the required package structure as well. Make sure to copy contents of the `ADBQuickstart.java` file to this new file.
3. Modify the `pom.xml` file with the following changes: Add Oracle JDBC Driver as a dependency.
`ojdbc8-production` will download Oracle JDBC driver (`ojdbc8.jar`) along with `ucp.jar` (required for using UCP as a client side connection pool), `orac1epki.jar`, `osdt_core.jar`,

osdt_cert.jar. These JARs are required for using Oracle Wallets while connecting to Oracle Autonomous AI Database.

```
<properties>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
</properties>
<dependencies>
  <dependency>
    <groupId>com.oracle.database.jdbc</groupId>
    <artifactId>ojdbc8-production</artifactId>
    <version>19.18.0.0</version>
    <type>pom</type>
  </dependency>
</dependencies>
```

4. Build and run a ADB QuickStart. Compile the Java code by right-clicking **ADBQuickStart.java**, and then click **Build Module ADBQuickStart**. Make sure you do not have any compilation error in the Java code and you are using the latest JDK version.
5. Run the sample Java program. Right-click **ADBQuickStart.java**, and then click **Run ADBQuickStart.main()**. Make sure to enter the database password on the console.
Sample Output:

The queried rows along with a success message are displayed, as shown in the following screen:



```
Enter the password for Autonomous Database:
Available connections after checkout: 4
Borrowed connections after checkout: 1

Query is SELECT CUST_ID, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_CITY, CUST_CREDIT_LIMIT FROM SH.CUSTOMERS
WHERE ROWNUM < 20 order by CUST_ID

CUST_ID CUST_FIRST_NAME CUST_LAST_NAME CUST_CITY CUST_CREDIT_LIMIT
-----
3228 Abigail Ruddy Hoofddorp 7000
4117 Abner Everett Clermont-1'Herault 15000
6783 Abigail Ruddy Schimmert 11000
7673 Abner Everett Schwaebisch Gmuend 11000
10338 Abigail Ruddy Scheveningen 1500
13894 Abigail Ruddy Joinville 9000
17449 Abigail Ruddy Nagoya 9000
21005 Abigail Ruddy Santos 3000
24561 Abigail Ruddy Yokohama 7000
25470 Abner Everett Stuttgart 15000
28116 Abigail Ruddy Haarlem 11000
31671 Abigail Ruddy Bolton 1500
35227 Abigail Ruddy Lelystad 9000
36117 Abner Everett Wolverhampton 15000
39672 Abner Everett Murnau 11000
43228 Abner Everett Los Angeles 7000
47006 Abner Everett Montara 11000
49671 Abigail Ruddy Ede 1500
50561 Abner Everett Neuss 7000

Congratulations! You have successfully used Oracle Autonomous Database

Available connections after checkin: 5
Borrowed connections after checkin: 0
```

Note

Note: If you connect to Oracle Autonomous AI Database from behind a firewall, you will likely encounter a connection timeout error. Make sure to be outside the firewall while running this sample or update the `tnsnames.ora` file to use an HTTPS proxy.

Gradle

1. Follow the instructions from the [Gradle Guide](#) for Gradle download and build instructions, and set the `PATH` variable before using Gradle commands. As a first step, create a Gradle project using the following command. Make sure to choose `2:application` for **Select type of project to generate**. Also, for **Source package (default:temp)**: use `com.oracle.jdbctest`.

```
gradle init
```

2. Copy the [ADBQuickStart.java](#) file to the `src/main/java/com/oracle/jdbc8test` directory.
3. Modify the `build.gradle` file with the following changes:
 - Add `mavenCentral()` as a repository.
 - Add Oracle JDBC driver as a dependency.

Note

`ojdbc8-production` downloads Oracle JDBC driver (`ojdbc8.jar`) along with `ucp.jar` (required for using UCP as a client side connection pool), `oraclepi.jar`, `osdt_core.jar`, `osdt_cert.jar`. These JARs are required for using Oracle Wallets while connecting to Oracle Autonomous AI Database.

For additional information, see [Maven Central Guide](#).

- Update the `mainClassName` to `ADBQuickStart`.
- Add a run block to read the password from the console.

```
repositories {  
    // Maven Central  
    mavenCentral()  
}  
dependencies {  
    // Get the 19.18.0.0 Oracle JDBC driver along with other companion  
    jars  
    implementation("com.oracle.database.jdbc:ojdbc8-production:19.18.0.0")  
}  
application {  
    // Define the main class for the application  
    mainClassName = '{your_project_directory}.ADBQuickStart'  
}  
// To pause to read the password from console  
run {  
    standardInput = System.in  
}
```

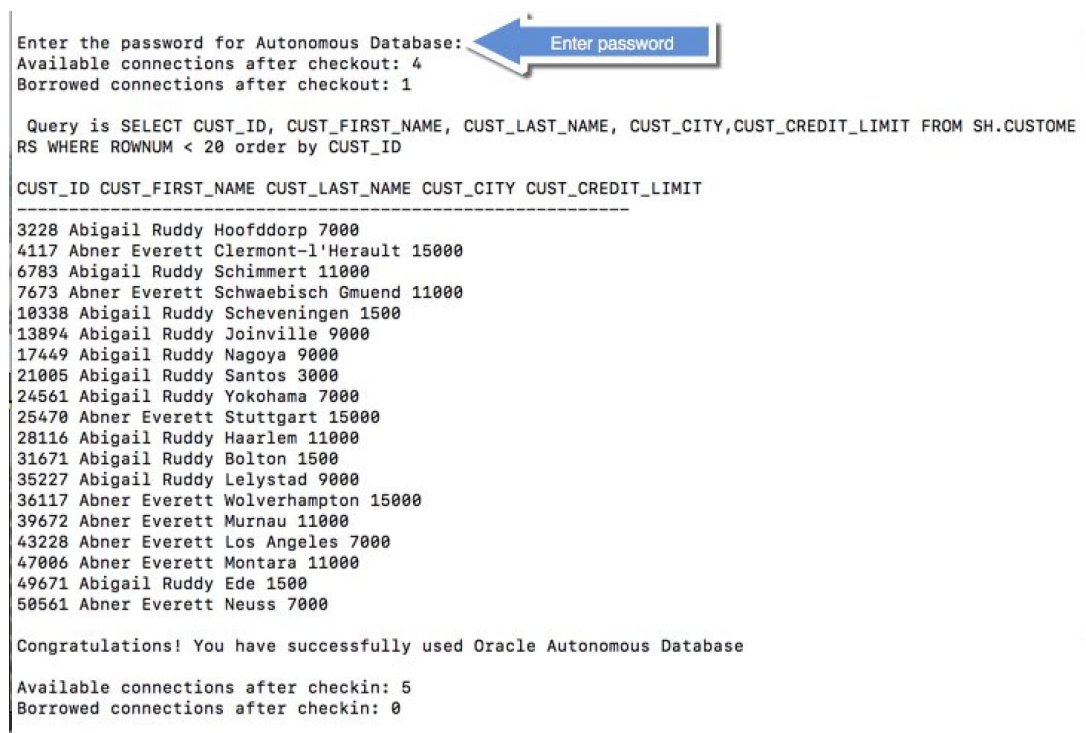
4. Build a Gradle Application. Make sure you are in the directory where build.gradle file is present. Compile the Java code using the following command:

```
./gradlew build
```

5. Run the sample Java program.

```
./gradlew run
```

Sample Output: The queried rows along with a success message are displayed, as shown in the following screen:



```

Enter the password for Autonomous Database:
Available connections after checkout: 4
Borrowed connections after checkout: 1

Query is SELECT CUST_ID, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_CITY, CUST_CREDIT_LIMIT FROM SH.CUSTOMERS WHERE ROWNUM < 20 order by CUST_ID

CUST_ID CUST_FIRST_NAME CUST_LAST_NAME CUST_CITY CUST_CREDIT_LIMIT
-----
3228 Abigail Ruddy Hoofddorp 7000
4117 Abner Everett Clermont-l'Herauld 15000
6783 Abigail Ruddy Schimmert 11000
7673 Abner Everett Schwaebisch Gmuend 11000
10338 Abigail Ruddy Scheveningen 1500
13894 Abigail Ruddy Joinville 9000
17449 Abigail Ruddy Nagoya 9000
21005 Abigail Ruddy Santos 3000
24561 Abigail Ruddy Yokohama 7000
25470 Abner Everett Stuttgart 15000
28116 Abigail Ruddy Haarlem 11000
31671 Abigail Ruddy Bolton 1500
35227 Abigail Ruddy Lelystad 9000
36117 Abner Everett Wolverhampton 15000
39672 Abner Everett Murnau 11000
43228 Abner Everett Los Angeles 7000
47006 Abner Everett Montara 11000
49671 Abigail Ruddy Ede 1500
50561 Abner Everett Neuss 7000

Congratulations! You have successfully used Oracle Autonomous Database

Available connections after checkin: 5
Borrowed connections after checkin: 0

```

Note

If you connect to Oracle Autonomous AI Database from behind a firewall, you will likely encounter a connection timeout error. Make sure to be outside the firewall while running this sample or update the `tnsnames.ora` file to use an HTTPS proxy.

Connect to Oracle Database Using JDBC Without a Build Tool

You can establish a connection to Oracle Autonomous AI Database manually without any build tool.

Prerequisites

The following sections provide information about the tasks that you need to perform before connecting your Java applications to Oracle Autonomous AI Database using Oracle JDBC driver and Universal Connection Pool. Refer to the appropriate prerequisites depending on the authentication type selected during provisioning of your Autonomous Exadata VM Cluster (AVMC). By default, one-way TLS connections are enabled when you provision an AVMC. See [Create an Autonomous Exadata VM Cluster](#) for more information.

You can find the authentication type on the Details page of your AVMC. See [View Details of an Autonomous Exadata VM Cluster](#) for instructions.

-
- [One-way TLS](#)
 - [mTLS](#)

One-way TLS

Provision an Oracle Autonomous AI Database Instance

You need access to an Oracle Autonomous AI Database. Refer to [Before You Begin with Autonomous AI Database on Dedicated Exadata Infrastructure](#) if you have no provisioned one already.

Remember the password that you used for ADMIN user. For the demonstration purpose, we will be using ADMIN user, but our recommendation is to create other database users either using Oracle SQL Developer or Database Actions.

Install JDK 8

Download latest [JDK 8](#) or a [higher JDK versions](#).

① Note

Ensure that you use JDK8u162 or a later version. Use `java -version` to check the JDK version that you have installed. To check the JDBC driver version, type `java -jar ojdbc8.jar`.

Download a Sample Program from Github

1. Download the [ADBQuickStart.java](#) file from Github. This sample application uses the Sales History (SH) sample schema and displays 20 records from the SH.CUSTOMERS table.
2. Modify the ADBQuickStart.java file to include your Oracle Autonomous AI Database connection information:
 - **DB_USER:** You can use ADMIN, the user created by default when the Oracle Autonomous AI Database is created (if you create another OracleAutonomous AI Database user, you can use that user instead).
 - **DB_PASSWORD:** Use the database user's password. If connecting as the ADMIN user, set this to the password you chose during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. For security reasons, you need to enter the password through the console when you run the sample.

- **DB_URL:** Obtain the connection string for your Autonomous AI Database by following the instructions in View Connection Strings for an Autonomous AI Database. If you are directly using in the Java program, you need to escape " in the connection string with \"

A sample quick start Java file looks like this:

```
DB_URL = "jdbc:oracle:thin:@jdbc:oracle:thin:@(description=
    (retry_count=20)(retry_delay=3)(address=(protocol=tcps)(port=1521)
    (host=adb.us-sanjose-1.oraclecloud.com))
    (connect_data=(service_name=g13ec47eade81f7_demodb_low.adb.oraclecloud.com)
    )(security=(ssl_server_cert_dn=\"CN=adb.us-sanjose-1.oraclecloud.com,
        OU=Oracle ADB SANJOSE, O=Oracle Corporation, L=Redwood City,
        ST=California,
        C=US\")))\"
DB_USER="ADMIN" and DB_PASSWORD="your_password"
```

3. Save changes to the ADBQuickStart.java file.

Troubleshooting and Debugging: If you encounter any error, refer the [Troubleshooting Tips](#) page for some helpful hints.

mTLS

Provision an Oracle Autonomous AI Database Instance

You need access to an Oracle Autonomous AI Database. Refer to Before You Begin with Autonomous AI Database on Dedicated Exadata Infrastructure if you have no provisioned one already.

Remember the password that you used for ADMIN user. For the demonstration purpose, we will be using ADMIN user, but our recommendation is to create other database users either using Oracle SQL Developer or Database Actions.

Obtain Client Credentials

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

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

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

Note

Protect the wallet.zip file and its contents to prevent unauthorized database access.

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

Install JDK 8

Download latest [JDK 8](#) or a [higher JDK versions](#).

Note

Ensure that you use JDK8u162 or a later version. Use `java -version` to check the JDK version that you have installed. To check the JDBC driver version, type `java -jar ojdbc8.jar`.

Download a Sample Program from Github

1. Download the [ADBQuickStart.java](#) file from Github. This sample application uses the Sales History (SH) sample schema and displays 20 records from the SH.CUSTOMERS table.
2. Modify the ADBQuickStart.java file to include your Oracle Autonomous AI Database connection information:
 - **DB_USER**: You can use ADMIN, the user created by default when the Oracle Autonomous AI Database is created (if you create another OracleAutonomous AI Database user, you can use that user instead).
 - **DB_PASSWORD**: Use the database user's password. If connecting as the ADMIN user, set this to the password you chose during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. For security reasons, you need to enter the password through the console when you run the sample.
 - **DB_URL**: Enter the net service name (TNS Alias) **DBName_medium** where **DBName** is the Oracle Autonomous AI Database Name entered during the **Create Autonomous AI Database** step while provisioning Oracle Autonomous AI Database. The available net service names can be seen in the `tnsnames.ora` file, which is a part of the client credentials zip file.
TNS_ADMIN should point to the location where you have unzipped the client credentials of Oracle Autonomous AI Database.

A sample quick start Java file looks like this:

```
DB_URL = "jdbc:oracle:thin:@DBName_medium?TNS_ADMIN=/Users/test/  
wallet_DBName" DB_USER="ADMIN" and DB_PASSWORD="enter_it_from_console"
```

3. Save changes to the ADBQuickStart.java file.

Procedure

1. Download the 19c version of [ojdbc8-full.tar.gz](#) from OTN and unzip the contents to your classpath.

Note

`ojdbc8-full.tar.gz` contains the latest JDBC driver `ojdbc8.jar`, `ucp.jar` (required JAR for using UCP as a client side connection pool), `oraclepki.jar`, `osdt_core.jar`, `osdt_cert.jar`. These JARs are required for using Oracle Wallets while connecting to Oracle Autonomous AI Database.

2. Compile the Java program. Make sure to provide the correct path for the required JARs in the classpath:

```
javac -classpath ./lib/ojdbc8.jar:./lib/ucp.jar:./lib/oraclepki.jar:./lib/osdt_core.jar:./lib/osdt_cert.jar com/oracle/jdbctest/ADBQuickStart.java
```

3. Run the sample Java program. Make sure to provide the correct path for the required JARs in the classpath:

```
java -classpath ./lib/ojdbc8.jar:./lib/ucp.jar:./lib/oraclepki.jar:./lib/osdt_core.jar:./lib/osdt_cert.jar:. com.oracle.jdbctest.ADBQuickStart
```

Sample Output:

The queried rows along with a success message are displayed, as shown in the following screen:



```
Enter the password for Autonomous Database:
Available connections after checkout: 4
Borrowed connections after checkout: 1

Query is SELECT CUST_ID, CUST_FIRST_NAME, CUST_LAST_NAME, CUST_CITY, CUST_CREDIT_LIMIT FROM SH.CUSTOMERS WHERE ROWNUM < 20 order by CUST_ID

CUST_ID CUST_FIRST_NAME CUST_LAST_NAME CUST_CITY CUST_CREDIT_LIMIT
-----
3228 Abigail Ruddy Hoofddorp 7000
4117 Abner Everett Clermont-l'Herauld 15000
6783 Abigail Ruddy Schimmert 11000
7673 Abner Everett Schwaebisch Gmuend 11000
10338 Abigail Ruddy Scheveningen 1500
13894 Abigail Ruddy Joinville 9000
17449 Abigail Ruddy Nagoya 9000
21005 Abigail Ruddy Santos 3000
24561 Abigail Ruddy Yokohama 7000
25470 Abner Everett Stuttgart 15000
28116 Abigail Ruddy Haarlem 11000
31671 Abigail Ruddy Bolton 1500
35227 Abigail Ruddy Lelystad 9000
36117 Abner Everett Wolverhampton 15000
39672 Abner Everett Murnau 11000
43228 Abner Everett Los Angeles 7000
47006 Abner Everett Montara 11000
49671 Abigail Ruddy Ede 1500
50561 Abner Everett Neuss 7000

Congratulations! You have successfully used Oracle Autonomous Database

Available connections after checkin: 5
Borrowed connections after checkin: 0
```

Note

If you connect to Oracle Autonomous AI Database from behind a firewall, you will likely encounter a connection timeout error. Make sure to be outside the firewall while running this sample or update the `tnsnames.ora` file to use an HTTPS proxy.

Secure Autonomous AI Database

Configure Zero Trust Packet Routing (ZPR) for an AVMC

APPLIES TO:  Oracle Public Cloud only

You can configure Zero Trust Packet Routing (ZPR) for an existing Autonomous Exadata VM Cluster (AVMC) using the Oracle Cloud Infrastructure console.

Required IAM Policies

```
allow group <group_name> to { ZPR_TAG_NAMESPACE_USE,  
SECURITY_ATTRIBUTE_NAMESPACE_USE } in tenancy  
allow group <group_name> to manage autonomous-database-family in tenancy  
allow group <group_name> to read security-attribute-namespaces in tenancy
```

Procedure

1. Go to the **Details** page of the AVMC for which you want to configure ZPR. See [View Details of an Autonomous Exadata VM Cluster](#).
2. Click **Security**. Click **Add security attribute**.

Note

You can also add security attributes by directly clicking **Add security attributes** under **Actions**.

3. In the **Add security attributes** dialog, select a **Namespace**, and enter a **Key** and **Value**.

Note

To select a security attribute at this step, you must already have set up security attributes with ZPR. For more information, see [About Zero Trust Packet Routing \(ZPR\)](#).

4. Save your changes.

Create and Manage Key Stores

You can create, view, edit, move, and delete Key Stores from the Oracle Cloud Infrastructure Console.

Create a Key Store

Follow these steps to create a Key Store to connect to an on-premises encryption key appliance such as Oracle Key Vault (OKV).

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.

For instructions, see [Access Dedicated Autonomous AI Database](#) in the Oracle Cloud Infrastructure Console.

2. Choose your Compartment.
3. Click **Key Stores**. Key Stores page displays the list name of key stores, the number of databases associated with each database, and the date on which each key store was created.
4. Click **Create Key Store**.
5. In the **Create Key Store** dialog, enter the following general information:
 - a. Name your key store: A user-friendly description or other information that helps you easily identify the Key Store resource. Avoid entering confidential information.
 - b. Oracle Key Vault connection settings:
 - i. **Connection IP addresses**: Enter at least one OKV cluster node IP address; multiple comma-separated IP addresses (of the same OKV cluster) are possible, for example, 193.10.20.1, 193.10.20.2.
 - ii. **Administrator username**: Enter the user name of the okv_rest_user.
 - iii. **Administrator Password Secret**: The administrator password is stored with the secret management service within OCI. Select the OCI Vault in your tenancy that contains okv_rest_user password stored as Secret.
 - c. Tags: Optionally, you can apply tags. If you have permission to create a resource, you also have permission to apply free-form tags to that resource. To apply a defined tag, you must have permission to use the tag namespace. For more information about tagging, see Tag Autonomous AI Database Cloud Resources. If you are not sure if you should apply tags, skip this option (you can apply tags later) or ask your administrator. Avoid entering confidential information.
6. Click **Create Key Store**.

View Key Store Details

Follow these steps to view Key Store details that include Oracle Key Vault (OKV) connection details and the list of associated databases.

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. Choose your Compartment.
3. Click **Key Stores**. Key Stores page displays the list name of Key Stores, the number of databases associated with each database, and the date on which each Key Store was created.
4. Click the name of the Key Store or click the **Actions** icon (three dots), and then click **View Details**.
5. Click the link in the Administrator Password Secret field to view secret details.

Edit Key Store Details

You can edit a Key Store only if it is not associated with any Autonomous Container Databases.

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.

2. Choose your Compartment.
3. Click **Key Stores**.
4. Click the name of the Key Store or click the Actions icon (three dots), and then click **View Details**.
5. On the Key Store Details page, click **Edit**.
6. On the Edit Key Store page, make changes as needed, and then click **Save Changes**.

Move a Key Store to Another Compartment

Follow these steps to move a Key Store from one compartment to another compartment.

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. Choose your Compartment.
3. Click **Key Stores**.
4. Click the name of the Key Store or click the **Actions** icon (three dots), and then click **View Details**.
5. On the **Key Store Details** page, click **Move Resource**.
6. On the **Move Resource to a Different Compartment** page, select the new compartment.
7. Click **Move Resource**.

View Key Store Associated Container Database Details

Follow these steps to view details of the Autonomous Container Database associated with a Key Store.

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. Choose your Compartment.
3. Click **Key Stores**.
4. In the **Key Stores** page, click the name of the Key Store or click the **Actions** icon (three dots), and then click **View Details**.
5. Click the name of the associated database or click the **Actions** icon (three dots), and then click **View Details**.

Delete a Key Store

You can delete a Key Store only if it is not associated with any Autonomous Container Databases.

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure Console.
For instructions, see Access Dedicated Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. Choose your Compartment.
3. Click **Key Stores**.

4. Click the name of the Key Store or click the **Actions** icon (three dots), and then click **View Details**.
5. On the **Key Store Details** page, click **Delete**.
6. On the **Delete Key Store** dialog, click **Delete**.

Prepare to Use Oracle Key Vault with Autonomous AI Database on Dedicated Exadata Infrastructure

Oracle Key Vault is a full-stack, security-hardened software appliance built to centralize the management of keys and security objects within the enterprise. The Oracle Key Vault is a customer-provisioned and managed system and it is not part of Oracle Cloud Infrastructure managed services. You can integrate your on-premises Oracle Key Vault (OKV) with customer-managed database cloud services to secure your critical data on-premises.

Prerequisites

1. Ensure that OKV is set up and the network is accessible from the Oracle Public Cloud client network. Open ports 443, 5695, and 5696 for egress on the client network to access the OKV server.
2. Ensure that the REST interface is enabled from the OKV user interface.
3. Create "OKV REST Administrator" user. You can use any qualified username of your choice, for example, "okv_rest_user". For Autonomous AI Database on Cloud@Customer and Oracle Database Exadata Cloud at Customer, use the same or different REST users. Those databases can be key-managed in the same or different on-prem OKV clusters. Oracle Database Exadata Cloud at Customer needs REST user with create endpoint privilege. Autonomous AI Database on Cloud@Customer needs REST user with create endpoint and create endpoint group privileges.
4. Gather OKV administrator credentials and IP address, which is required to connect to OKV and configure the key store. See [Create a Key Store](#) for guidance.
5. Open the ports 443, 5695, and 5696 for egress on the client network to access the OKV server.
6. On Oracle Public Cloud deployments, ensure OKV has network access to the Autonomous AI Database by setting proper network routes with VPN (Fast connect or VPN as a Service) or any VCN peering if the compute host is in another VCN.

Create a Vault in OCI Vault Service and Add a Secret to the Vault to Store OKV REST Administrator Password

Your Dedicated Exadata Infrastructure deployment communicates with OKV over REST each time an Oracle Database is provisioned to register the Oracle Database and request a wallet on OKV. Therefore, Exadata infrastructure needs access to the REST admin credentials to register with the OKV server. These credentials are stored securely in the Oracle Vault Service in OCI as a Secret and accessed by your Dedicated Exadata Infrastructure deployment only when needed. When needed, the credentials are stored in a password-protected wallet file.

Create a Policy Statement for Database Service to Use Secret from OKV Vault Service

To grant the Autonomous AI Database service permission to use the secret in OCI Vault to log in to the OKV REST interface, navigate to (or create) an IAM policy in a compartment higher up in your compartment hierarchy than the compartment containing your OCI Vaults and Secrets. Then, add a policy statement of this format:

```
allow service database to read secret-family in compartment <vaults-and-secrets-compartment>
```

where <vaults-and-secrets-compartment> is the name of the compartment in which you created your OCI Vaults and Secrets.

Once the OCI Vault is set up and the IAM configuration is in place, you are now ready to deploy your Oracle Key Vault 'Key Store' in OCI and associate it with your Dedicated Exadata VM Cluster.

Update OKV Endpoint Group

You can update the OKV endpoint group from the Details page of an ACD.

Optionally, you can also add an OKV endpoint group name with the OKV keystore while provisioning the ACD or later.

To update the OKV endpoint group on an ACD, you must ensure that:

- The OKV endpoint group has access to the corresponding OKV wallet.
- The OKV endpoints corresponding to the ACD are part of the OKV Endpoint group.
- The OKV endpoint group has read access to default wallet(s) of the endpoints.

To update the OKV endpoint group name:

- Go to the **Details** page of the ACD. For instructions, see [View Details of an Autonomous Container Database](#).
- Click **Edit** next to the OKV Endpoint Group name under **Encryption**.
- Choose a Key Store from the list and enter a name for the OKV endpoint group. The endpoint group name must be in all uppercase and may include numbers, hyphens (-), and underscores (_) and start with an uppercase letter.
- Click **Save**.

Manage OKV Endpoints

Delete Oracle Key Vault Endpoints

After you terminate an ACD, you should delete the endpoints corresponding to the ACD from OKV. OKV endpoints are created at the time of ACD provisioning per ACD per cluster node. Deleting the endpoints corresponding to a terminated ACD keeps the OKV organized and helps during OKV CA certificate rotation.

Deleting an endpoint removes it permanently from Oracle Key Vault. However, security objects that were previously created or uploaded by that endpoint will remain in Oracle Key Vault. Similarly, security objects that are associated with that endpoint also remain. To permanently delete or reassign these security objects, you must be a user with the Key Administrator role or

authorized to merge these objects by managing wallet privileges. The endpoint software previously downloaded at the endpoint also remains on the endpoint until the endpoint administrator removes it.

You cannot delete an endpoint that is in the `PENDING` state unless you are the user who created it. You must delete it on the node on which it was created. See [Deleting One or More Endpoints](#) for more details.

Re-enrolling Endpoints

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure does not support Re-enrolling OKV endpoints out of the box. To re-enroll OKV endpoints, you need to contact the Autonomous AI Database operations teams.

Register or Deregister a Dedicated Autonomous AI Database with Data Safe

To use Oracle Data Safe, you register your database with Oracle Data Safe. To discontinue using it, you deregister your database.



Tip

For a "try it out" alternative that demonstrates using Data Safe with Autonomous AI Database, run the *labs 2 to 13* in Oracle Autonomous AI Database Dedicated for Security Administrators Workshop.

Before You Begin

Before you can register your database with Data Safe, Data Safe must be configured to access databases in your dedicated infrastructure configuration, as described in [Create an Oracle Data Safe Private Endpoint](#). Usually this one-time configuration is performed by the fleet administrator of the dedicated infrastructure configuration in coordination with the network administrator of the tenancy.

If Database Vault is enabled on the database you are registering or deregistering, you need to perform special steps to accommodate the additional database security provided by Database Vault:

- To register or deregister, the Database Vault Account Manager must first grant specific access rights to the `ADMIN` database user, as described in [Register an Autonomous AI Database on Dedicated Exadata Infrastructure](#). After the registration or deregistration operation completes, the Database Vault Account Manager can revoke these rights and so restore the Database Vault configuration to its previous state.
- After registering, the Database Vault Owner must grant specific access rights to the `DS$ADMIN` database user, as described in [Grant Roles to the Oracle Data Safe Service Account on Your Autonomous AI Database](#).

Procedure

1. Go to the **Details** page of the Autonomous AI Database you want to register or deregister with Oracle Data Safe.

For instructions, see [View Details of a Dedicated Autonomous AI Database](#).

2. On the **Details** page, under **Data Safe**, click **register** or **deregister** depending on the status of the database.

A confirmation dialog box is displayed.

3. Enter the password of the ADMIN database user and then click **Confirm** to start the registration or deregistration operation.

You can track completion of the operation using its work request, as described in Use Work Requests to Monitor Long-Running Operations.

Rotate Encryption Keys

You can rotate the master encryption keys associated with an Autonomous AI Database on Dedicated Exadata Infrastructure using the Oracle Cloud Infrastructure Console.

Rotate the Encryption Key of an Autonomous Container Database

Required IAM Policies

`manage autonomous-container-databases`

Procedure

1. Go to the **Details** page of the Autonomous Container Database whose encryption key you want to rotate.
For instructions, see View Details of an Autonomous Container Database.
2. Under **Actions**, click **Rotate Encryption Key**.
3. (Optional) To use a customer encryption key (BYOK), select **Rotate using the customer-provided key (BYOK)**. BYOK is supported in Oracle Public Cloud only.
 - **For External KMS:** Each third-party key is automatically assigned a key version in the external HSM.
 - Rotate the third-party keys in the external HSM so that the external HSM generates a new key version.
 - Copy the version ID of the rotated key and use it to rotate the key reference in OCI Key Management (EKMS) so that OCI Key Management (EKMS) can create a new Key version OCID.
 - Copy the newly created Key Version OCID from EKMS.
 - **For OCI Vaults:** Enter the OCID of the imported customer encryption key in **Key Version OCID**. The Key Version OCID that you enter should be associated with the current encryption key of the Autonomous Container Database.
4. Click **Rotate encryption Key**.

The Autonomous Container Database goes to the **Updating** status, the encryption key is rotated, and the Autonomous Container Database goes back to the **Active** status. How the encryption key is rotated depends on whether it is Oracle-managed or customer-managed :

- **Oracle-managed key:** Autonomous AI Database rotates the encryption key, storing the new value in the secure key store on the Exadata system where the Autonomous Container Database resides.
- **Customer-managed key:** Autonomous AI Database uses the underlying technology (Oracle Cloud Infrastructure Vault for Autonomous Container Databases on Oracle Public Cloud and Multicloud deployments, or Oracle Key Vault (OKV) for Autonomous Container Databases on either Oracle Public Cloud or Exadata Cloud@Customer, or AWS KMS for Autonomous AI Database on Oracle Database@AWS) to rotate the key and store the new

value as a new version of the key in underlying technology, and then associates this new version with the Autonomous Container Database.

Rotating the AWS KMS key generates a new encryption context for the same key.

You can view the latest Key Version OCID and the entire Key History from your Autonomous Container Database details page. This is not applicable for AWS KMS keys.

Note

In case of cross region Data Guard with Customer Managed Keys, the replicated vault used by the standby is read-only. So, when the standby assumes the primary role from a failover, you cannot rotate the key.

Rotate the Encryption Key of an Autonomous AI Database

You rotate the encryption key of an Autonomous AI Database from its **Details** page.

1. Go to the **Details** page of the Autonomous AI Database whose encryption key you want to rotate.
For instructions, see View Details of a Dedicated Autonomous AI Database.
2. On Oracle Public Cloud, click **Rotate Encryption Key** under **More actions**, and on Exadata Cloud@Customer, click **Rotate Encryption Key** under **Actions**.
3. (Optional) To use a customer encryption key (BYOK), select **Rotate using the customer-provided key (BYOK)**. BYOK is supported in Oracle Public Cloud only.
 - **For External KMS:** Each third-party key is automatically assigned a key version in the external HSM.
 - Rotate the third-party keys in the external HSM so that the external HSM generates a new key version.
 - Copy the version ID of the rotated key and use it to rotate the key reference in OCI Key Management (EKMS) so that OCI Key Management (EKMS) can create a new Key version OCID.
 - Copy the newly created Key Version OCID from EKMS.
 - **For OCI Vaults:** Enter the OCID of the imported customer encryption key in **Key Version OCID**. The Key Version OCID that you enter should be associated with the current encryption key of the Autonomous Container Database.
4. Click **Rotate encryption Key**.

The Autonomous AI Database goes to the **Updating** status, the encryption key is rotated, and the Autonomous AI Database goes back to the **Active** status. How the encryption key is rotated depends on whether it is Oracle-managed or customer-managed:

- **Oracle-managed key:** Autonomous AI Database rotates the encryption key, storing the new value in the secure key store on the Exadata system where the Autonomous AI Database resides.
- **Customer-managed key:** Autonomous AI Database uses the underlying technology (Oracle Cloud Infrastructure Vault for Autonomous Container Databases on Oracle Public Cloud and Multicloud deployments, or Oracle Key Vault (OKV) for Autonomous Container Databases on either Oracle Public Cloud or Exadata Cloud@Customer, or AWS KMS for Autonomous AI Database on Oracle Database@AWS) to rotate the key and store the new value as a new version of the key in underlying technology, and then associates this new version with the Autonomous Container Database.
Rotating the AWS KMS key generates a new encryption context for the same key.

You can view the latest Key Version OCID and the entire Key History from your Autonomous Container Database details page. This is not applicable for AWS KMS keys.

Note

In case of cross region Data Guard with Customer Managed Keys, the replicated vault used by the standby is read-only. So, when the standby assumes the primary role from a failover, you cannot rotate the key.

Set Access Control List for an Autonomous AI Database

An Access Control List (ACL) provides additional protection to your Autonomous AI Database by allowing only the client with specific IP addresses to connect to the database. You can add IP addresses individually, or in CIDR blocks. Enabling an ACL with an empty list of IP addresses makes the database inaccessible.

1. Go to the **Details** page of the Autonomous AI Database to which you want to set access control.

For instructions, see [View Details of a Dedicated Autonomous AI Database](#).

2. In the **Autonomous AI Database Information** tab, locate **Access Control List** under **Network** and click **Edit**.

Editing an ACL is enabled only when the database state is **Available**.

Note

For Autonomous Data Guard enabled databases, the **Access Control List** for both the primary and standby databases can be edited *only* from the primary database. The **Details** page of a standby database allows you *only* to view the ACLs.

3. The **Edit Access Control List** dialog appears with existing ACLs and their values.

Optionally, edit an existing ACL by overwriting its values or remove it completely by clicking **X** next to it.

Note

If **Access Control List** is disabled for this Autonomous AI Database, you do not see any ACL to edit. Click **Enable database level access control** to enable ACL.

4. Optionally, add an ACL by selecting or entering basic information:
 - a. Click **+ Access Control Rule**.
 - b. Depending on the types of addresses in your list, select one of the following options from the **IP notation type** drop-down selector:
 - **IP Address:** Specify individual IP addresses. You can use commas to separate multiple IP addresses.

Note

An IP address specified in a network ACL entry should be the public IP address of the client that is visible on the public internet that you want to grant access. For example, for an Oracle Cloud Infrastructure VM, this is the IP address shown in the **Public IP** field on the Oracle Cloud Infrastructure console for that VM.

- **CIDR Block:** Specify ranges of public IP addresses using CIDR notation. You can use commas to separate multiple values.
5. For Autonomous Data Guard enabled databases, you see options to define access control for the standby database. Depending on your preference, select one of the following options for **Standby database access control**:
- Choose **Same as primary database** to apply the ACL that you already defined for the primary database to the standby database as well. This is the default selection.

Note

By default, the ACL defined for the primary database applies to the standby database also.

- Choose **Define standby database access control** to define a separate ACL for the standby database. By default, the standby database ACL is initialized with the same values as the primary database ACL. Add or modify the entries as needed.
6. Click **Save**.
- If the **Lifecycle State** is **Available** when you click **Save** the **Lifecycle State** changes to **Updating** until the ACL is set. The database is still up and accessible, there is no downtime. When the update is complete the **Lifecycle State** returns to **Available** and the network ACLs from the access control list are in effect.

Create and Manage Autonomous AI Database Software Images

Create an Autonomous AI Database Software Image

Autonomous AI Database software images are resources within your tenancy that give you the ability to create a customized database software configuration.

You can create an Autonomous AI Database software image from the Oracle Cloud Infrastructure console. The maximum number of Autonomous AI Database software images that can be created in a compartment is 10.

Note

The lifecycle state of an Autonomous AI Database software image changes to **Expired** when its Container Database software version is two versions or more, behind the latest RU.

Required IAM Policies

manage auto-db-software-img

- [From Autonomous AI Database Overview Page](#)
- [From Autonomous Container Database Details Page](#)

From Autonomous AI Database Overview Page

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure console.
For instructions, see [Access Autonomous AI Database in the OCI Console](#).
2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous AI Database software images**.
The list of Autonomous AI Database software images in your current Compartment is displayed.
3. Choose the **Compartment** where you want to see the list of Autonomous AI Database software image. The list of Autonomous AI Database software images refreshes to show those in the selected Compartment.
4. Click **Create Autonomous AI Database software image**.
5. Fill out the **Create Autonomous AI Database software image** page with the following information:

Setting	Description	Notes
Basic information	Compartment: Select a compartment to host the Autonomous AI Database software image. Display Name: Enter a user-friendly description or other information that helps you easily identify the Autonomous AI Database software image.	Avoid entering confidential information.
Existing Autonomous Container Database	Region: Select the underlying region for the Autonomous Container Database (ACD). Exadata Infrastructure: Select the underlying Exadata Infrastructure resource for the ACD. Autonomous Exadata VM Cluster: Select the underlying Autonomous Exadata VM Cluster resource for the ACD. Autonomous Container Database: Select the ACD to create the software image from.	-
Show/hide advanced options	By default, advanced options are hidden. Click Show advanced options to display them.	-

Setting	Description	Notes
Advanced options: Tags	If you want to use tags, add tags by selecting a Tag Namespace , Tag Key , and Tag Value .	Tagging is a metadata system that allows you to organize and track resources within your tenancy. See Tagging Autonomous AI Database Cloud Resources.

6. Click **Create**.

From Autonomous Container Database Details Page

1. Navigate to the details page of the ACD you want to create a software image from.
2. Click **Create image from database** under **Actions**.
3. On the **Create Autonomous AI Database software image**, select a compartment and enter a display name for the software image.
4. By default, advanced options are hidden. Click **Show advanced options** to display them.
5. Optionally, add tags by selecting a **Tag Namespace**, **Tag Key**, and **Tag Value**.
6. Click **Create**.

Manage Autonomous AI Database Software Images

You can manage an Autonomous AI Database software image from the Oracle Cloud Infrastructure console.

View a List of Autonomous AI Database Software Images

You can view a list of Autonomous AI Database software images in a compartment using the Oracle Cloud Infrastructure console.

Required IAM Policies

`inspect auto-db-software-img`

Procedure

1. Go to Autonomous AI Database in the Oracle Cloud Infrastructure console.
For instructions, see Access Autonomous AI Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous AI Database software images**.

The Autonomous AI Database software images page is displayed, showing the list of Autonomous AI Database software images in your current Compartment. Choose the **Compartment** where you want to see the list of Autonomous AI Database software images.

The following details are displayed for each Autonomous AI Database software image in the list:

- State of the resource such as Updating, Provisioning, Available, etc.

- Database version of the Autonomous AI Database software image
- Date and time of creation of the resource.

View Details of an Autonomous AI Database Software Image

Follow these steps to view detailed information about an Autonomous AI Database software image.

Required IAM Policies

`inspect auto-db-software-img`

Procedure

1. Go to **Autonomous AI Database** in the Oracle Cloud Infrastructure console.
For instructions, see [Access Autonomous AI Database](#) in the Oracle Cloud Infrastructure Console .
2. In the side menu's list of Autonomous AI Database resource types, click **Autonomous AI Database software images**.
The Autonomous AI Database software images page is displayed, showing the list of Autonomous AI Database software images in your current Compartment. Choose the **Compartment** where you want to see the list of Autonomous AI Database software images.
3. Click the name of the Autonomous AI Database software image whose details you want to view.

The **Details** page for the chosen Autonomous AI Database software image is displayed.

Actions Allowed

You can perform a variety of actions on the **Details** page for an Autonomous AI Database software image.

Action	Steps
Move Autonomous AI Database software image to a different compartment	Click Move Resource . On the Move Resource to a Different Compartment dialog, choose the new compartment and click Move Resource .
Add tags to the Autonomous AI Database software image	Click Add Tags .
Terminate the Autonomous AI Database software image	Click Delete and follow the instructions in Delete an Autonomous AI Database Software Image .
View or copy Autonomous AI Database software image's OCID	On the Autonomous AI Database software image information tab, click Show or Copy in the OCID field.
View or copy one-off patches	On the Autonomous AI Database software image information tab, click Show or Copy in the One-off patches field.

Resources Listed

As you scroll down, the following resources associated with this Autonomous AI Database software image are listed on the Autonomous AI Database software imagedetails page:

- **Work Requests:** All the work requests associated with this Autonomous AI Database software image are listed along with their state and other details.

Delete an Autonomous AI Database Software Image

Follow these steps to terminate an Autonomous AI Database software image.

Required IAM Policies

manage auto-db-software-img

Procedure

1. Go to the **Details** page of the Autonomous AI Database software image that you want to move. See [View Details of an Autonomous AI Database Software Image](#).
2. Click **Delete**.
3. Confirm that you wish to terminate your Autonomous AI Database software image in the confirmation dialog.
4. Click **Delete**.

Create and Manage Elastic Pools

Create an Elastic Pool

You can create an elastic pool using an existing Autonomous AI Database instance, while provisioning an Autonomous AI Database, or while cloning an Autonomous AI Database.

Note

Only Autonomous AI Transaction Processing databases that use ECPU compute model without Autonomous Data Guard can create an elastic pool.

Create an Elastic Pool Using an Existing Autonomous AI Database

Follow these steps to create an elastic pool using the Oracle Cloud Infrastructure console.

1. Go to the **Details** page of the Autonomous AI Database using which you want to create an elastic pool.

For instructions, see [View Details of an Autonomous AI Database](#).

2. On Oracle Public Cloud, click **Manage resource allocation** under **More actions**, and on Exadata Cloud@Customer, click **Manage resource allocation** under **Actions**.
3. On the **Manage resource allocation** dialog, verify if **Compute auto scaling** is disabled. If Compute auto scaling is enabled already, disable it by *deselecting* **Compute auto scaling** and clicking **Apply**.

The lifecycle state changes to **Scaling in Progress**. After some time the lifecycle state changes to **Available**.

4. Once the Autonomous AI Database lifecycle state changes to **Available**, click **Manage resource allocation** again.
5. Click **Show advanced options**.

6. In the **Elastic pool** section:
 - a. Select the **Enable elastic pool** check box.
 - b. Select the **Create an elastic pool** option.
 - c. Select a pool size for the **Pool ECPU count** field from available list of pool shapes: 128, 256, 512, 1024, 2048, or 4906.
7. Click **Apply**.

The Lifecycle state changes to **UPDATING**. The changes apply immediately after the Lifecycle state changes to **Available**.

After you create an elastic pool, click **Manage resource allocation** to display the elastic pool information. In the **Elastic pool** section, you can see **Elastic pool** as Enabled, **Pool role** as Leader, and **Pool ECPU count** same as what you selected while creating the elastic pool.

Create an Elastic Pool While Provisioning or Cloning an Autonomous AI Database

Follow these steps to create an elastic pool when you provision or clone an Autonomous AI Database instance:.

1. Launch the **Create Autonomous AI Database** or **Clone Autonomous AI Database** page following the steps documented in [Create an Autonomous AI Database](#) or [Clone an Autonomous AI Database](#).
2. In the **Configure the database** section:
 - a. Deselect **Compute auto scaling**.
 - b. Click **Show advanced options**.
 - c. In the **Elastic pool** section, select the **Enable elastic pool** check box.
 - d. Select the **Create an elastic pool** option.
 - e. Select a pool size for the **Pool ECPU count** field from available list of pool shapes: 128, 256, 512, 1024, 2048, or 4906.
3. Continue with the remaining steps in creating or cloning the Autonomous AI Database.

Join or Leave an Elastic Pool

Autonomous AI Database instances that are added to an existing pool are pool members. As a pool member, you can join or leave an existing elastic pool.

Note

Elastic pools are only available for Autonomous AI Database instances that use the ECPU compute model without Autonomous Data Guard.

Join an Elastic Pool

To join an elastic pool:

1. Go to the **Details** page of the Autonomous AI Database.
For instructions, see [View Details of an Autonomous AI Database](#).
2. On Oracle Public Cloud, click **Manage resource allocation** under **More actions**, and on Exadata Cloud@Customer, click **Manage resource allocation** under **Actions**.

3. On the **Manage resource allocation** dialog, verify if **Compute auto scaling** is disabled. If Compute auto scaling is enabled already, disable it by *deselecting* **Compute auto scaling** and clicking **Apply**.

The lifecycle state changes to **Scaling in Progress**. After some time the lifecycle state changes to **Available**.

4. Once the Autonomous AI Database lifecycle state changes to **Available**, click **Manage resource allocation** again.
5. Click **Show advanced options**.
6. In the **Elastic pool** section:
 - a. Select the **Enable elastic pool** check box.
 - b. Select the **Join an existing elastic pool** option.
 - c. Choose the pool leader from the **Select pool leader in compartment** select list.
7. Click **Apply**.

The Lifecycle state changes to **UPDATING**. The changes apply immediately after the Lifecycle state changes to **Available**.

Join an Elastic Pool While Provisioning or Cloning an Autonomous AI Database

To join an elastic pool **while provisioning or cloning** an Autonomous AI Database:

1. Launch the **Create Autonomous AI Database** or **Clone Autonomous AI Database** page following the steps documented in *Create an Autonomous AI Database* or *Clone an Autonomous AI Database*.
2. In the **Configure the database** section:
 - a. Select the **Enable elastic pool** checkbox.
 - b. Select the **Join an existing elastic pool** option.
 - c. Select a region from the **Region** select list, if you want to join an elastic pool whose pool leader is in another region. The default value is current region.
 - d. Choose the pool leader from the **Select pool leader in compartment** select list.
3. Continue with the remaining steps in creating or cloning the Autonomous AI Database.

Leave an Elastic Pool Autonomous AI Database

As a pool member, you can remove yourself from an elastic resource pool. See **Notes for Leaving an Elastic Resource Pool** in the Elastic Resource Pool Billing topic for the billing impact when a pool member leaves an elastic pool.

To leave an elastic pool:

1. Go to the **Details** page of the Autonomous AI Database which you want to remove from the elastic pool.

For instructions, see *View Details of an Autonomous AI Database*.
2. Locate the **Resource allocation** section on Oracle Public Cloud and the **Resources** section on Exadata Cloud@Customer. Click **Leave pool** in the **Elastic pool** field.

The **Leave pool** confirmation dialog displays.
3. Click **Leave**.

The Lifecycle state changes to **UPDATING**. The changes apply immediately after the Lifecycle state changes to **Available**.

Manage an Elastic Pool as a Pool Leader

The Autonomous AI Database instance that creates an elastic pool is the pool leader. A pool leader can perform the following operations on an elastic pool.

Change an Elastic Pool Shape

Only a pool leader can change the elastic pool shape for an existing elastic pool.

To change the shape of an elastic pool (update the pool size):

1. Go to the **Details** page of the Autonomous AI Database that is a pool leader.
For instructions, see View Details of an Autonomous AI Database.
2. On Oracle Public Cloud, click **Manage resource allocation** under **More actions**, and on Exadata Cloud@Customer, click **Manage resource allocation** under **Actions**.
3. Click **Show advanced options**.
4. In the **Elastic pool** section, select a value that is different than the current value for the **Pool ECPU count** field.
5. Click **Apply**.

The Lifecycle state changes to **UPDATING**. The changes apply immediately after the Lifecycle state changes to **Available**.

Note

Decreasing the CPU allocation, Pool ECPU count, to a value that cannot accommodate all the members of the elastic pool is not allowed.

For example, for an elastic pool with a Pool ECPU count of 256 ECPUs and a pool capacity of 1024 ECPUs: If the elastic pool contains eight (8) Autonomous AI Database instances with 80 ECPUs each for a total of 640 ECPUs, the elastic pool leader cannot decrease the Pool ECPU count to 128 ECPUs. In this case, if the pool size were reduced to 128 ECPUs, the pool capacity would be 512 ECPUs, which is less than the total allocation for the pool members (640 ECPUs).

List Elastic Pool Members

Follow these steps to list the members of an elastic pool:

1. Go to the **Details** page of the Autonomous AI Database that is a pool leader.
For instructions, see View Details of an Autonomous AI Database.
2. Click **Elastic pool members**.
This shows the **Elastic pool members** section, with a list showing details for each instance that is an elastic pool member.
3. If you click the **ellipses (three vertical dots)** at the end of a row in the list, you can select an action to perform for the member. The possible actions are:

- **View details:** Shows the member's Oracle Cloud Infrastructure Console.
- **Copy OCID:** Copies the member's Autonomous AI Database instance OCID.
- **Remove from pool:** Brings up a dialog where you can confirm to remove the Autonomous AI Database instance from the pool.

Remove an Elastic Pool Member

Follow these steps to remove an elastic pool member, as a pool leader:

1. Go to the **Details** page of the Autonomous AI Database that is a pool leader.
For instructions, see [View Details of an Autonomous AI Database](#).
2. Click **Elastic pool members**.
This shows the **Elastic pool members** section, with a list showing details for each instance that is an elastic pool member.
3. Click the **ellipses (three vertical dots)** at the end of the row for an instance you want to remove, and select **Remove from pool** from the drop-down list.
This shows the **Remove from pool** confirmation dialog.
4. Click **Leave** to confirm.

The Lifecycle state changes to **UPDATING**. The changes apply immediately after the Lifecycle state changes to **Available**.

Terminate an Elastic Pool

Only the pool leader can terminate an elastic pool.

Note

Terminating an elastic pool is only allowed when there are no pool members in the elastic pool.

To terminate an elastic pool:

1. Go to the **Details** page of the Autonomous AI Database that is a pool leader.
For instructions, see [View Details of an Autonomous AI Database](#).
2. On Oracle Public Cloud, click **Manage resource allocation** under **More actions**, and on Exadata Cloud@Customer, click **Manage resource allocation** under **Actions**.
3. Click **Show advanced options**.
4. In the **Elastic pool** section, select the **Terminate pool** check-box.
5. Click **Apply**.

The Lifecycle state changes to **UPDATING**. The changes apply immediately after the Lifecycle state changes to **Available**.

View a List of Remote Backups

Remote backups, also known as cross region backups, are additional backups that are stored in a different region than the source database. In case of a region failure, you can use this backup to recover your database. When you enable cross-region backups, a backup will be

taken at the Autonomous Container Database (ACD) level. This backup includes the Autonomous AI Databases within the ACD and all archive logs. A second backup will be taken in the region that you've specified. The ACD backup can then be cloned into a different Autonomous Exadata VM Cluster (AVMC). You can view a list of remote backups from the Oracle Cloud Infrastructure console.

To create a remote backup, you must enable cross region backups for your Autonomous Container Database (ACD). See [Edit Autonomous Container Database Backup Settings](#) for instructions.

Prerequisites

`autonomous_container_database_backup_inspect`

Procedure

1. Go to **Autonomous AI Database** in the Oracle Cloud Infrastructure console.
2. Click **Remote backups** under Autonomous AI Database resource types from the side menu.

The **Remote backups** page displays, showing the list of remote backups in your current compartment. You can optionally change the compartment to view a list of remote backups in that compartment.

Note

After creating a remote backup, you can see the Remote backup region from the ACD Details page. You can also click this link to open the list of remote backups.

You can see the list of the following details for each Remote backup in the list:

- Name of the resource
- State of the resource such as Updating, Provisioning, Available, etc.
- Resource type. Each remote backup will be an ACD.

7

Reference

Previous Feature Announcements

New Feature Announcements

Announcements for the noteworthy changes made to Autonomous AI Database on Dedicated Exadata Infrastructure.

December 2025

Feature	Description
AWR Report on Standby Databases	With Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can access an Automatic Workload Repository (AWR) report on a standby Autonomous AI Database. The AWR report collects, processes, and maintains performance statistics for problem detection and self-tuning purposes. This report can be accessed through Performance Hub on the Details page of your standby database. See Performance Monitoring and Tuning Tools and Observe Autonomous AI Database on Dedicated Exadata Infrastructure with Performance Hub for details.
Cross region clone of an Autonomous AI Database S3 backups	Using Autonomous AI Database on Oracle Database@AWS, you can perform a cross region clone from an Autonomous AI Database S3 backups. That is you can clone an Autonomous AI Database from an S3 backup stored in one AWS region into a different Autonomous Container Database in a different AWS region. See Clone an Autonomous AI Database for more details.
Use DBMS_CLOUD_NOTIFICATION package to send OCI Notifications	The DBMS_CLOUD_NOTIFICATION package allows you to send messages to OCI Notifications Service. When you use the OCI Notifications service with DBMS_CLOUD_NOTIFICATION package, only the DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE procedure is supported (the DBMS_CLOUD_NOTIFICATION.SEND_DATA procedure is not supported). See DBMS_CLOUD_NOTIFICATION Package for more details.

November 2025

Feature	Description
Select AI Support for AWS	With Autonomous AI Database on Dedicated Exadata Infrastructure deployments, Select AI supports integration with AWS. See Configure AI Profiles Using DBMS_CLOUD_AI, Example: Select AI with AWS, and DBMS_CLOUD_AI Package Profile Attributes for more information.
Retain the terminated database S3 backups and lock the ACD backup retention period	Autonomous AI Database on Dedicated Exadata Infrastructure does not delete the database S3 backups upon terminating any (or all) of the Autonomous AI Database components, like Exadata Infrastructure, Autonomous VM Cluster (AVMC), ACD, and Autonomous AI Database. You can enable the backup retention lock to prevent any unwanted deletion of the ACD backups and long-term backups before their retention period after an ACD is terminated. See About Backup and Recovery for more details.
Undelete a terminated Autonomous AI Database on Oracle Database@AWS	You can undelete an Autonomous AI Database on Oracle Database@AWS within 72 hours of its termination, provided the corresponding ACD is not terminated. See Undelete an Autonomous AI Database for more details.
New critical and informational events for Autonomous Exadata VM Cluster, Autonomous Container Database, and Autonomous AI Database	With Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can now get notified when database services and nodes are online and offline, and when FRA usage in an Autonomous Container Database is 90% or more. See Events for Autonomous AI Database on Dedicated Exadata Infrastructure for more details.
New critical events for Autonomous Container Database and Autonomous AI Database.	With Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can now get notified when the Database Vault (DV) is disabled on an Autonomous AI Database instance by any means other than Oracle's standard, approved patching automation. See Events for Autonomous AI Database on Dedicated Exadata Infrastructure for more details.
DBMS_CLOUD_FUNCTION Package Support	With Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can use the DBMS_CLOUD_FUNCTION package to invoke OCI and AWS Lambda remote functions in your Autonomous AI Database as SQL functions. See the following pages for more information: • DBMS_CLOUD_FUNCTION Package • DBMS_CLOUD_FUNCTION Views

Feature	Description
Use In-database Transformer Model with Select AI RAG	Autonomous AI Database on Dedicated Exadata Infrastructure supports in-database transformers (also known as embedding models) that you imported into your Oracle AI Database 26ai instance with Select AI RAG to generate vector embeddings for document chunks and user prompts. See Use In-database Transformer Models and Example: Select AI with In-database Transformer Models Example: Select AI with In-database Transformer Models for more information.
Use Select AI to Generate Synthetic Data	Autonomous AI Database on Dedicated Exadata Infrastructure can use Select AI to generate synthetic data using LLMs that conforms to your schema for solution testing, proofs of concept, and other uses. Synthetic data can be used for better testing of your applications when real data is not available and can also be used to populate an Autonomous AI Database clone or a metadata clone. See Synthetic Data Generation for more information.

October 2025

Feature	Description
Cross-Tenancy Standby database using Autonomous Data Guard Groups model	In Oracle Public Cloud deployments of Autonomous AI Database on Dedicated Exadata Infrastructure, when you create an Autonomous Container Database, you can create up to two standby databases in a different tenancy from the primary database using the Autonomous Data Guard Groups model. This choice allows you to add a standby database from a different tenancy, letting your database failover or switchover to that cross-tenancy standby database. See Autonomous Data Guard Configuration Options and Add a Cross Tenancy Standby Database for more details.

Feature	Description
Oracle Autonomous AI Database	Oracle AI Database architects Artificial Intelligence (AI) into its core, seamlessly integrating AI across all major data types and workloads. See the Press Release for details. As a result, Oracle Autonomous Database becomes Oracle Autonomous AI Database. Additionally, Oracle Autonomous Transaction Processing is Oracle Autonomous AI Transaction Processing, and Oracle Autonomous JSON Database is Oracle Autonomous AI JSON Database. Oracle Autonomous AI Lakehouse has been added as the next generation of Autonomous Data Warehouse. It retains all functionality while introducing a host of capabilities for tightly integrating with Iceberg and other open-source technologies. See the Press Release for details. Refer to Autonomous AI Lakehouse for an overview. Oracle Autonomous AI Database 26ai replaces Oracle Autonomous Database 23ai. As a result, you can continue to use CreateAutonomousDatabase with 23ai for dbVersion. The REST API returns Oracle Database 23ai, but you will see Oracle Database 26ai in the Oracle Cloud Infrastructure (OCI) console. You can notice the new service names in the Oracle Cloud Infrastructure (OCI) console, REST API, and related documentation. See Autonomous AI Database for more details.
Additional log support for OCI Logging Service	In addition to the available logs on Autonomous AI Database on Dedicated Exadata Infrastructure, the following logs are available to be used with OCI Logging Service: • ORDS Connection Logs for monitoring operations related to connecting your Autonomous AI Database to Oracle REST Data Services. • Mongo Connection Logs for monitoring operations related to connecting to your Autonomous AI Database via Oracle Database API for MongoDB. • Network Logs for monitoring network activity. See Observe Resource Logs with OCI Logging Service for more details.

September 2025

Feature	Description
Retain the terminated database backups and lock the ACD backup retention period	Autonomous AI Database on Dedicated Exadata Infrastructure does not delete the database backups upon terminating any (or all) of the Autonomous AI Database components, like Exadata Infrastructure, Autonomous VM Cluster (AVMC), ACD, and Autonomous AI Database. You can enable the backup retention lock to prevent any unwanted deletion of the ACD backups and long-term backups before their retention period after an ACD is terminated. See About Backup and Recovery for more details.
Undelete a terminated Autonomous AI Database	You can undelete an Autonomous AI Database within 72 hours of its termination, provided the corresponding ACD is not terminated. See Undelete an Autonomous AI Database for more details.
Directory File Support for Data Pipelines and AI	You can use directory files, including those in network file systems, with data pipelines in Autonomous AI Database on Dedicated Exadata Infrastructure. All functions available with object storage files are supported with directory files. Refer the following pages for more information: - Create and Configure Pipelines - DBMS_CLOUD_PIPELINE Package
Wild Card File Selection in Autonomous AI Database APIs	You can use wild card patterns to select multiple files in both object storage and directory objects when working with DBMS_CLOUD, DBMS_CLOUD_PIPELINE, and DBMS_CLOUD_AI in Autonomous AI Database on Dedicated Exadata Infrastructure. Refer the following pages for more information: - DBMS_CLOUD_PIPELINE Package - DBMS_CLOUD for Objects and Files - DBMS_CLOUD for Bulk File Management
Query external data on Oracle Compute Cloud at Customer (C3) Object Storage Bucket	You can use the DBMS_CLOUD package to configure Autonomous AI Database deployments on Exadata Cloud@Customer deployments to load, query, and manage unstructured data stored on C3 Object Storage Bucket. See Query External Data on C3 Object Storage Bucket for more details.

August 2025

Feature	Description
Oracle Key Vault (OKV) endpoint group association with Autonomous Container Database (ACD)	With Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can associate a new or existing Autonomous Container Database (ACD) to a custom OKV endpoint group. See Create an Autonomous Container Database and Update Oracle Key Vault Endpoint Group for instructions.
DBMS_EXT_TABLE_CACHE Package Support	With Autonomous AI Database on Dedicated Exadata Infrastructure deployments on Oracle Database 23ai or later, you can use the DBMS_EXT_TABLE_CACHE package to cache frequently accessed data from external tables in your database. This can provide improved performance where queries run much faster for frequently accessed external data. See the following pages for more information: • Improve Query Performance with External Table Cache • DBMS_EXT_TABLE_CACHE Package
Query Apache Iceberg Tables	Autonomous AI Database on Dedicated Exadata Infrastructure supports querying of Apache Iceberg tables stored in Amazon Web Services (AWS) or in Oracle Cloud Infrastructure (OCI) Object Storage. See Query Apache Iceberg Tables for more information.
Support for CS_SESSION Package	With Autonomous AI Database on Dedicated Exadata Infrastructure, you can use the CS_SESSION package to switch the database service and consumer group of the existing session. See CS_SESSION Package for more information.

July 2025

Feature	Description
Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@AWS	Using Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Database@AWS, you can run Oracle Database workloads on Oracle Cloud Infrastructure (OCI) managed Exadata Infrastructure within AWS public regions and Availability Zones. Applications with Autonomous AI Lakehouse, Autonomous AI Transaction Processing, or mixed workloads can seamlessly connect to Autonomous AI Databases running on Exadata Infrastructure deployed in AWS regions. The Autonomous AI Database on Oracle Database@AWS has native integration with AWS services such as AWS S3, CloudWatch (metrics, logs), EventBridge, and other service capabilities like AWS ZeroETL. See Autonomous AI Database on Dedicated Exadata Infrastructure in AWS region for details.
Documentation Update: New events for Autonomous Exadata VM Cluster and Autonomous Container Database.	With Autonomous AI Database on Dedicated Exadata Infrastructure deployments, you can now get notified when: • One of the diskgroups for your Autonomous Exadata VM Cluster is almost out of space. • The lack of free space in one of the diskgroups for your Autonomous Exadata VM Cluster has been resolved. • Your Autonomous Container Database has an issue that requires investigation by Operations. Refer to Events for Autonomous AI Database on Dedicated Exadata Infrastructure for a complete list of events emitted by Autonomous AI Database resources.

June 2025

Feature	Description
New experience in Oracle Cloud Console	Autonomous AI Database on Dedicated Exadata Infrastructure launched a major update to the Oracle Cloud Console. The new experience includes intuitive designs across the Console to enhance usability and consistency, with significant updates to resource list pages, resource details pages, and resource creation workflows. Learn more about the new Console experience.

Feature	Description
MongoDB API support from the Oracle Cloud Infrastructure (OCI) console	You can enable the MongoDB API on your Autonomous AI Database on Dedicated Exadata Infrastructure from the OCI console. The MongoDB API enables MongoDB-compatible clients and drivers to connect directly to your Autonomous AI Database. See Use MongoDB API with Oracle Autonomous AI Database for instructions on enabling the MongoDB API and connecting MongoDB applications to an Autonomous AI Database.
Documentation Update: Autonomous AI Database on Exadata Cloud@Customer Events	Autonomous AI Database on Dedicated Exadata Infrastructure documentation is updated with the complete list of events emitted by the following Exadata Cloud@Customer resources such as Exadata Infrastructure, Autonomous Exadata VM Cluster, Autonomous Container Database, Autonomous AI Database. You can refer to Events for Autonomous AI Database on Dedicated Exadata Infrastructure to view the updated events.
Oracle Application Express (APEX) 24.2 is available for new Autonomous Container Databases (ACD)	You can use Oracle Application Express 24.2 to develop applications for Autonomous AI Database on Dedicated Exadata Infrastructure. Only newly created ACDs have access to Oracle Application Express 24.2. All existing ACDs will continue to run with the APEX version they were provisioned with. To use the latest Oracle Application Express features for enhancing the existing applications, you can schedule an upgrade to your APEX instance through Service Request (SR) in My Oracle Support. You must supply the OCIDs of the tenancy and Autonomous AI Database for which you request the APEX upgrade in the SR. See Build an Oracle Application Express (APEX) Application for more details.
Documentation Update: New events for Autonomous Container Database	With Autonomous AI Database deployments on Exadata Cloud@Customer, you can now get notified when: - Your Network File Storage server is out of space. - Incorrect permissions are set on your Network File Storage server. - Your Zero Data Loss Recovery Appliance server is out of space. - Your Network File Storage backups consume at least 80% of space in the backup destination. - Your Autonomous Container Database's backup destination is back to a healthy state. Refer to Events for Autonomous AI Database on Dedicated Exadata Infrastructure for a complete list of events emitted by Autonomous AI Database resources.

Feature	Description
Ability to create a cross region backup copy of an Autonomous Container Database	Oracle Public Cloud deployments of Autonomous AI Database on Dedicated Exadata Infrastructure allow you to select a secondary region for a copy of your Autonomous Container Database (ACD) backups. You can also view and clone remote backups from the OCI console. In case of a region failure, you can clone the ACD's remote backup to restore the data. The backup of an ACD can only be cloned to an Autonomous Exadata VM Cluster (AVMC) different from its source AVMC. For instructions on enabling cross region backup copies, see Create an Autonomous Container Database and Edit Autonomous Container Database Backup Settings.
Ability to clone an Autonomous Container Database	On Oracle Public Cloud deployments, Autonomous AI Database on Dedicated Exadata Infrastructure lets you clone an Autonomous Container Database (ACD) and all its Autonomous AI Databases from the ACD Details page on the Oracle Cloud Infrastructure (OCI) console. See About Cloning Autonomous Container Database on Dedicated Exadata Infrastructure and Clone an Autonomous Container Database for further information and guidance.

May 2025

Feature	Description
OCPU billing metric retirement for new Autonomous Exadata VM Clusters (AVMCs)	Starting May 28th, 2025, Autonomous AI Database on Dedicated Exadata Infrastructure does not allow you to create new AVMCs using the OCPU billing metric. All the new AVMCs can only be provisioned with ECPUs. Existing OCPU AVMCs and Autonomous AI Databases will continue to operate as usual. You can have your OCPU AVMCs and respective Autonomous AI Databases updated to ECPU via service requests. See Doc ID 2998755.1 for more information.
Transport Layer Security 1.3 (TLS 1.3) certificate-based authentication	Autonomous AI Database on Dedicated Exadata Infrastructure with Oracle Database version 23ai or later can use either TLS 1.2 or TLS 1.3 (Transport Layer Security) certificate-based authentication for database connections. See Encryption of Data in Transit

Feature	Description
Use custom kerberos service name for Kerberos authentication	Autonomous AI Database on Dedicated Exadata Infrastructure supports custom kerberos service name for Kerberos authentication. When you use a custom name you must specify the <code>kerberos_service_name</code> parameter with <code>DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION</code>. Specifying this parameter is not required when you use the GUID for the service name. See Components of the Kerberos Authentication System for more details.
New database metric for Session Utilization	Autonomous AI Database on Dedicated Exadata Infrastructure provides a new metric called <code>SessionUtilization</code> to show the maximum session utilization as a percentage value, aggregated across all consumer groups. See Available Metrics: <code>oci_autonomous_database</code> for more details.

April 2025

Feature	Description
Manage Customer Contacts for Autonomous AI Database resources	When customer contacts are set, Oracle sends announcements, operational notifications, and unplanned maintenance notifications to the specified email addresses. With Autonomous AI Database on Dedicated Exadata Infrastructure, you can add, view, and update customer contacts at the following resource levels: • Exadata Infrastructure • Autonomous Container Database (ACD) • Autonomous AI Database See the following pages for more instructions: • Manage Customer contacts for an Exadata Infrastructure • Create an Autonomous Container Database • Manage Customer contacts for an Autonomous Container Database • Clone an Autonomous AI Database Instance • Clone an Autonomous AI Database from Backup

March 2025

Feature	Description
Provision Autonomous Exadata VM Clusters with IPv4/IPv6 dual-stack networking	In Oracle Public Cloud deployments of Autonomous AI Database on Dedicated Exadata Infrastructure, you can provision Autonomous Exadata VM Clusters with IPv4/IPv6 dual-stack networking, enabling both IPv4 and IPv6 addresses. This option is enabled while you create a VCN, and while creating a subnet. You can use the corresponding subnet (which has both IP4 and IP6 addresses) while provisioning an Autonomous Exadata VM Cluster.
Integration with OCI Logging Service	With Autonomous AI Database on Dedicated Exadata Infrastructure, you can enable or disable capturing of Oracle Cloud Infrastructure service logs using the OCI Logging Service. See Observe Resource Logs with OCI Logging Service for details.
Select AI with Google, Anthropic, and Hugging Face	Autonomous AI Database on Dedicated Exadata Infrastructure can interact with AI service providers. In addition to the previously supported providers, Select AI supports Google Gemini, Anthropic, and Hugging Face. This feature supports having LLMs work with Oracle database by generating SQL from natural language prompts, enabling you to talk to your database. See Use Select AI to Generate SQL from Natural Language Prompts for information on using Select AI with Autonomous AI Database.
Dual standby database support in an Autonomous Data Guard configuration	Autonomous AI Database on Dedicated Exadata Infrastructure lets you enable Autonomous Data Guard from the Details page of an Autonomous Container Database (ACD) and create up to two standby ACDs. The second standby ACD must be in the same tenancy as the primary ACD. See Add a Second Standby Autonomous Container Database for instructions. With this release, the previous <i>Autonomous Data Guard Associations</i> model and associated APIs will be deprecated and replaced with the new <i>Autonomous Data Guard Groups</i> model and APIs. All new ACDs provisioned after March 2025 from the Oracle Cloud Infrastructure (OCI) console will automatically use the new <i>Autonomous Data Guard Groups</i> model. See Knowledge Base for more information. To transition existing ACDs, customers can migrate to the new model by clicking the Upgrade to Autonomous Data Guard Groups from the ACD Details page on the OCI Console or using the MigrateAutonomousContainerDatabaseDataguardAssociation API. See Autonomous Data Guard Models to know more details about upgrading to the new model.

February 2025

Feature	Description
NFS space utilization	With Autonomous AI Database deployments on Exadata Cloud@Customer, if the backup destination type is NFS, then you can view the NFS space utilization from the ACD details page. The current NFS space utilization is displayed as a percentage along with a status icon. See View NFS Space Utilization for more details.
Change the backup destination type of an Autonomous Container database	With Autonomous AI Database deployments on Exadata Cloud@Customer, you can change the backup destination type for an Autonomous Container Database which has the automatic backup enabled. See Edit Autonomous Container Database Backup Settings for more details.
Restore Database to a specific SCN	On Autonomous AI Database on Dedicated Exadata Infrastructure, you can restore the database to a specific System Change Number (SCN). See Restore and Recover Autonomous AI Database on Dedicated Exadata Infrastructure for more details.
Database Clone from the latest backup	On Autonomous AI Database on Dedicated Exadata Infrastructure, when you clone the database from a backup, you can choose the Backup clone type as Latest backup timestamp . With this backup clone type, you can clone the database from the most recent backup data that is available. See Clone Autonomous AI Database from Backup for more details.
Oracle Maintained Pipelines	Autonomous AI Database on Dedicated Exadata Infrastructure provides built-in pipelines that enable you to export the database audit logs or Oracle APEX workspace activity log to OCI Object Storage in JSON format. These pipelines are preconfigured and can be started by the ADMIN user. See Oracle Maintained Pipelines for more information.

January 2025

Feature	Description
Doc update: Vulnerability Assessment and Security Testing Policies	We have updated the Autonomous AI Database on Dedicated Exadata Infrastructure documentation with details about: • Vulnerability scans and patching that Autonomous AI Database frequently performs using commercial vulnerability scanning tools. • Security testing policies governing activities such as penetration testing and vulnerability scanning that Oracle customers can perform against their Autonomous AI Database. See Vulnerability Scans and Patching and Oracle Cloud Security Testing Policy for complete details.
Support for DELETE_OPERATION procedure in DBMS_CLOUD package.	On Autonomous AI Database on Dedicated Exadata Infrastructure, you can use <code>DBMS_CLOUD.DELETE_OPERATION</code> procedure to clear the data load entries for the specified operation ID logged in the <code>user_load_operations</code> or <code>dba_load_operations</code> tables in your schema.
X11M systems available for Autonomous AI Database on Oracle Public Cloud and Autonomous AI Database on Exadata Cloud@Customer deployments.	When you create Autonomous AI Database deployments on Oracle Public Cloud or Exadata Cloud@Customer, you can deploy to X11M Exadata Database Machine systems. For details about the capacities and characteristics of the X11M systems supported by the Autonomous AI Database, see Exadata X11M Systems.

New Feature Announcements (2024)

Announcements for the noteworthy changes made to Autonomous AI Database on Dedicated Exadata Infrastructure in 2024.

December 2024

Feature	Description
My Oracle Cloud Support	My Oracle Cloud Support is a redesigned support portal that replaces My Oracle Support (MOS), providing a unified sign-in experience with your Cloud Account. If you are using Autonomous AI Database on Dedicated Exadata Infrastructure in a commercial cloud region, you can now sign in directly with your Cloud Account credentials to access the new portal. Oracle has already migrated all your CSIs and open SRs to the new support portal, and you can continue to work on these SRs. - If you had an account in My Oracle Support before the migration, you can continue to use that account username and password to sign in to My Oracle Cloud Support. However, we do recommend that you use your Cloud Account instead. - You can access My Oracle Support to review your closed SRs as needed. Please see the following articles for guidance: - Register to My Oracle Cloud Support as a New User - Create a Service Request in My Oracle Cloud Support
Bring Your Own Keys (BYOK) on Oracle Public Cloud deployments	With Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Public Cloud, when you create a customer-managed key or key version, you can import your own key material instead of letting the Vault service generate the key material internally. See Use Bring Your Own Keys (BYOK) in the Vault Service for more details.
Use OCI External Key Management System (EKMS) Keys with Oracle Public Cloud deployments.	You can use a third-party on-premises encryption key from OCI EKMS with Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Public Cloud. See Use External Keys from OCI External Key Management Service (OCI EKMS) for more details.
Doc update: Overview of an Exadata Infrastructure (EI)	Our documentation has been updated with a new article to provide a complete overview of Exadata Infrastructure. This article answers all questions about Exadata Infrastructure, including what EI is, the requirements for creating EI, which database features can be defined or managed using EI, and provides related documentation to help you create and manage EI. See About Exadata Infrastructure for details.

November 2024

Feature	Description
Availability of OCI Full Stack Disaster Recovery (FSDR) on Autonomous AI Database on Dedicated Exadata Infrastructure	Full Stack Disaster Recovery provides full application recovery and minimizes disaster recovery time, providing recovery for the entire application stack instead of individual resources. You can enable the OCI Full Stack Disaster Recovery and use it to perform switchover/failover operations or optionally perform database only Autonomous AI Database switchover/failover operations. See Use OCI Full Stack Disaster Recovery on Autonomous AI Database on Dedicated Exadata Infrastructure for more details.
Connect to Oracle Autonomous AI Database using JDBC with or without a build tool	You can connect your Java applications to Oracle Autonomous AI Database using Oracle JDBC driver. Both one-way TLS and mutual TLS (mTLS) connections are available. You can connect with JDBC using Eclipse or IntelliJ, both while using either Maven or Gradle. Alternatively, you can connect manually without a build tool. See Connect to Oracle Autonomous AI Database Using JDBC with Eclipse, Connect to Oracle Autonomous AI Database Using JDBC with IntelliJ, and Connect to Oracle Database Using JDBC Without a Build Tool for instructions.
Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases	You can use <code>DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK</code> to create database links with <i>customer-managed heterogeneous connectivity</i> from an Autonomous AI Database on Dedicated Exadata Infrastructure to: - Publicly Accessible Non-Oracle Databases - Non-Oracle Databases on a private Endpoint, with or without a wallet. Refer to the following pages to know the prerequisites and steps involved in creating customer-managed heterogeneous connectivity: Create Database Links with Customer-Managed Heterogeneous Connectivity to Publicly Accessible Non-Oracle Databases Create Database Links with Customer-Managed Heterogeneous Connectivity to Non-Oracle Databases on a Private Endpoint

October 2024

Feature	Description
Support for regular expressions in DBMS_CLOUD procedures	On Autonomous AI Database on Dedicated Exadata Infrastructure, you can use the <code>regexuri</code> format option with the following DBMS_CLOUD procedures: • <code>COPY_COLLECTION</code> • <code>COPY_DATA</code> • <code>COPY_DATA</code> for Avro, ORC, or Parquet files • <code>CREATE_EXTERNAL_TABLE</code> • <code>CREATE_EXTERNAL_TABLE</code> for Avro, ORC, or Parquet files • <code>CREATE_EXTERNAL_PART_TABLE</code> • <code>CREATE_HYBRID_PART_TABLE</code> For more information, see DBMS_CLOUD Package Format Options and DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.
Support for X10M Base System	When you create Autonomous AI Database deployments on Exadata Cloud@Customer, you can deploy to X10M Exadata Database Machine Base system. See Exadata X10M Systems for system specifications.
Ability to create a cross tenancy Autonomous Data Guard Standby Database	On Oracle Public Cloud deployments, you can add an Autonomous Data Guard standby database that resides in a tenancy different from the primary database. The cross tenancy standby database can be in the same region or a different region (cross region). A cross tenancy standby database can only be added using the CLI or the Autonomous AI Database REST APIs. This option is not available using the Oracle Cloud Infrastructure console. See Add a Cross Tenancy Standby Database for instructions.
Updated Object Storage URL Format using Dedicated Endpoints for commercial (OC1) realms	Autonomous AI Database on Dedicated Exadata Infrastructure supports Oracle Cloud Infrastructure Object Storage URI formats for commercial (OC1) realms. See Cloud Object Store URI Formats for more information.

Feature	Description
Configure Zero Trust Packet Routing (ZPR) for Autonomous AI Database on Dedicated Exadata Infrastructure deployments on Oracle Public Cloud.	Oracle Cloud Infrastructure Zero Trust Packet Routing (ZPR) protects sensitive data from unauthorized access through intent-based security policies you write for resources, such as an Autonomous Exadata VM Cluster (AVMC) to which you assign security attributes. Security attributes are labels that ZPR uses to identify and organize resources. ZPR enforces policy at the network level each time access is requested, regardless of potential network architecture changes or misconfigurations. ZPR is built on the existing network security group (NSG) and security control list (SCL) rules. For a packet to reach a target, it must pass all NSG and SCL rules and ZPR policy. The request is dropped if any NSG, SCL, or ZPR rule or policy doesn't allow traffic. For Autonomous AI Database deployments on Oracle Public Cloud, you can apply ZPR security attributes to an AVMC while provisioning it or later. See the following pages for more information: • About Zero Trust Packet Routing (ZPR) • Create an Autonomous Exadata VM Cluster • Configure Zero Trust Packet Routing (ZPR) for an AVMC
Export data to Cloud Object Storage as Parquet.	Use <code>DBMS_CLOUD.EXPORT_DATA</code> to export data as text. The text format export options are CSV, JSON, Parquet, or XML. See Export Data to Object Store as Text Using DBMS_CLOUD.EXPORT_DATA for more information.

September 2024

Feature	Description
Availability of new Dynamic Performance Views on Autonomous AI Database on Dedicated Exadata Infrastructure.	You can use Dynamic Performance Views to: • View real-time metrics on different wait events and wait classes. • View historical snapshots of wait class metrics. • View real-time, historical, and summary performance metrics data. • View resource limits and current usage. See Database Observability with Dynamic Performance Views for additional information.

August 2024

Feature	Description
Select AI with OCI Generative AI	Autonomous AI Database can interact with AI service providers. Select AI now supports OCI Generative AI and Azure OpenAI Service. This feature supports having LLMs work with Oracle database by generating SQL from natural language prompts. This allows you to talk to your database. See Use Select AI to Generate SQL from Natural Language Prompts for more information.
DBMS_PIPE Support	The DBMS_PIPE package lets two or more sessions in the same instance communicate is available in Autonomous AI Database. Singleton Pipe is an addition to the DBMS_PIPE package that allows you to cache a custom message. Using a Singleton Pipe you can send and retrieve a custom message and share the message across multiple database sessions with concurrent reads. The DBMS_PIPE package also supports persistent messaging, where messages are stored in Cloud Object Store. See Cache Messages with Singleton Pipes, Use Persistent Messaging with Messages Stored in Cloud Object Store, DBMS_PIPE Package (Singleton Pipes), and DBMS_PIPE Package (Persistent Messaging Pipes) for more information.
Ability to update an Autonomous Container Database using a Custom database software image.	You can choose to update your Autonomous Container Database using an existing custom database software image with rolling or non-rolling maintenance methods. See Schedule a Quarterly Maintenance Update for additional information.
Version-enable one or more user tables using Oracle Workspace Manager.	Oracle Workspace Manager provides an infrastructure that enables applications to create workspaces and group different versions of table row values in different workspaces. Use Oracle Workspace Manager to version-enable one or more user tables in the database. See Using Oracle Workspace Manager on Autonomous AI Database for more information.

July 2024

Feature	Description
Elastic Pools support	On Oracle Public Cloud and Exadata Cloud@Customer deployments, you can use an elastic pool to consolidate your Autonomous AI Database instances, in terms of their allocation of compute resources, and to provide up to 87% cost savings. Refer to the following articles to know more about elastic pools, their terminology, billing, and related tutorials: • Consolidate Autonomous AI Database Instances Using Elastic Pools • Elastic Pool Billing • Create an Elastic Pool • Join or Leave an Elastic Pool • Manage an Elastic Pool as a Pool Leader
Ability to encrypt or decrypt data while exporting or importing data to or from Object Store.	For greater security you can encrypt data that you export to Object Storage. When data on Object Storage is encrypted you can decrypt the data you import or when you use the data in an external table. Encryption and decryption during import and export can be performed using DBMS_CRYPT or using a user defined function. See Encrypt Data While Exporting to Object Storage and Decrypt Data While Importing from Object Storage for additional information.
Cross tenancy cloning support on Oracle Public Cloud deployments.	On Oracle Public Cloud deployments, you can clone an Autonomous AI Database instance or its backup from one tenancy (source tenancy) to a different tenancy (destination tenancy). The cloned database can be in the same region or in a different region (cross-region) and both full and metadata cloning support cross tenancy cloning. The cross tenancy cloning option is only available using the CLI or the Autonomous AI Database REST APIs. This option is not available using the Oracle Cloud Infrastructure Console. See the following articles for more information: • Cross Tenancy Clone Requirements • Cross Tenancy Clone an Autonomous AI Database • Cross Tenancy Clone an Autonomous AI Database from Backup

Feature	Description
Doc update: Overview of an Autonomous AI Database	Our documentation has been updated with a new article to provide a complete overview of Autonomous AI Database resources. This article answers all questions about Autonomous AI Databases, including what an Autonomous AI Database is, the requirements for creating an Autonomous AI Database, which database features can be defined or managed using the Autonomous AI Database resource, the tools available to Autonomous AI Database users, and provides related documentation to help you create and manage Autonomous AI Databases. See About Autonomous AI Database.

June 2024

Feature	Description
Customer-managed keys in Oracle Key Vault (OKV) for Oracle Public Cloud deployments of Autonomous AI Database on Dedicated Exadata Infrastructure.	You can integrate your on-premises Oracle Key Vault (OKV) with Autonomous AI Database on Dedicated Exadata Infrastructure deployed on Oracle Public Cloud and use the customer-managed keys stored in Oracle Key Vault to secure your critical data. See Create Autonomous Container Database for details. Before you can use customer-managed keys stored in Oracle Key Vault, you must perform a number of preparatory configuration tasks as described in Prepare to Use Oracle Key Vault.
Ability to create and manage Key Stores.	You can create Key Stores using the Oracle Cloud Infrastructure (OCI) console to integrate your on-premises Oracle Key Vault deployment with Oracle Autonomous AI Database on Dedicated Exadata Infrastructure. These Key Stores let you create and manage your own the master keys using Oracle Key Vault. See Create and Manager Key Stores for instructions.

Feature	Description
Identity and Access Management (IAM) Integration with Autonomous AI Database on Exadata Cloud@Customer.	You can configure an Autonomous AI Database on Exadata Cloud@Customer to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous AI Database with IAM credentials. As of this release, (IAM) authentication and authorization: • Will be enabled on all the newly provisioned Autonomous Container Databases (ACDs) and Autonomous AI Databases by default. • Cannot be used with Autonomous AI Databases configured with Autonomous Data Guard. See Use Identity and Access Management (IAM) Authentication with Autonomous AI Database for more details.
Azure Active Directory (Azure AD) Integration with Autonomous AI Database on Exadata Cloud@Customer.	Azure Active Directory (Azure AD) users can now connect to an Autonomous AI Database deployed on Exadata Cloud@Customer using Azure OAuth2 access tokens. See Authenticate and Authorize Microsoft Azure Active Directory Users for Autonomous AI Database for more information.

May 2024

Feature	Description
23ai database support on Autonomous AI Database on Dedicated Exadata Infrastructure	You can provision 23ai Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Public Cloud and Exadata Cloud@Customer deployments , allowing you to take advantage of all the new features that Oracle Database 23ai offers. 23ai database is currently supported with the following limitations: • 23ai databases can be created only on Autonomous Exadata VM Clusters (AVMCs) with ECPUs, created on or after the launch of 23ai support, with the appropriate Tags. See 23ai Database Software Version Tag Requirements for details. • Autonomous AI Databases with 23ai software version can not be cloned into an Autonomous AI Database with 19c version and vice-versa.

Feature	Description
Support for Database Links from Autonomous AI Database on Dedicated Exadata Infrastructure to another Autonomous AI Database	You can create TLS and Non-TLS database links from Autonomous AI Database on Dedicated Exadata Infrastructure to a target Autonomous AI Database using the DBMS_CLOUD_ADMIN package sub-programs. Refer to the following articles for more information: - Create TLS Database Links from an Autonomous AI Database to another Autonomous AI Database - Create Non-TLS Database Links from an Autonomous AI Database to another Autonomous AI Database - DBMS_CLOUD_ADMIN Package Reference
Support for Database Links from Autonomous AI Database on Dedicated Exadata Infrastructure to Oracle Database.	You can create database links from Autonomous AI Database on Dedicated Exadata Infrastructure to a target Oracle database that is either on a public endpoint or a private endpoint using the DBMS_CLOUD_ADMIN package sub-programs. Refer to the following articles for more information: - Create Database Links from an Autonomous AI Database to a Publicly Accessible Oracle Database with a Wallet (mTLS) - Create Database Links from an Autonomous AI Database to an Oracle Database on a Private Endpoint - DBMS_CLOUD_ADMIN Package Reference
Doc update: Overview of an Autonomous Exadata VM Cluster (AVMC)	Our documentation has been updated with a new article to provide a complete overview of Autonomous Exadata VM Cluster resources. This article answers all questions about Autonomous Exadata VM Clusters, including what an AVMC is, the requirements for creating an AVMC, which database features can be defined or managed using the AVMC resource, and provides related documentation to help you create and manage AVMCs. See About Autonomous Exadata VM Cluster.
Network File System (NFS) NFSv4 Support	You can attach external NFS devices provided by the customer on an NFSv4 server to Autonomous AI Database, using the DBMS_CLOUD_ADMIN procedures. Depending on the version of the Network File System you want to access, both NFSv3 and NFSv4 are supported. Refer to the following for more information: - Attach Network File Storage to Autonomous AI Database - Attach External File Storage to Autonomous AI Database - DBMS_CLOUD_ADMIN Package

Feature	Description
Amazon S3 Presigned URLs	You can use a presigned URL in any DBMS_CLOUD procedure that takes a URL to access files in Amazon Simple Storage Service, without the need to create a credential. See Amazon S3 Compatible URI Format for more information.

April 2024

Feature	Description
Doc update: Use case demonstration to configure Autonomous AI Database on Dedicated Exadata Infrastructure for proof of concept (POC) applications.	Our documentation is updated with a use case demonstrating how to quickly configure your Autonomous AI Database resources on Dedicated Exadata Infrastructure for developing proof of concept (POC) applications. If you are looking to quickly create an Autonomous AI Database POC environment, see Use Case: Configure Autonomous AI Database for Proof of Concept (POC).
Doc Update: Instructions to request a service limit increase.	We have added a new article that demonstrates how to submit a request to increase your service limits using the Oracle Cloud Infrastructure console to add Exadata Infrastructure resources - Database Servers and Storage Servers to the tenancy. See Request a Service Limit Increase for details.
View and manage customer contacts for operational issues and announcements.	When customer contacts are set, Oracle sends announcements, operational notifications, and unplanned maintenance notifications to the specified email addresses. You can add email contacts while provisioning Autonomous AI Databases. You can also view and modify customer contacts for an existing Autonomous AI Database using the Oracle Cloud Infrastructure console. See Manage Customer Contacts for more information.
Ability to provision an Autonomous Container Database using the Oracle Database software version of an Autonomous AI Database software image .	You can choose the latest Oracle Database software version, its immediate predecessor, or a custom database software image as the Container Database software version while provisioning an Autonomous Container Database. See Create an Autonomous Container Database for instructions.
Ability to create Autonomous AI Database software images.	You can create Autonomous AI Database software images and use them as standard images for provisioning Autonomous Container Databases. See Create an Autonomous AI Database software image for instructions.

Feature	Description
Ability to view and copy a list of one-off patches available for an Autonomous Container Database.	You can view and copy a list of one-off patches from the Autonomous Container Database Details page. See Service Maintenance of Autonomous AI Database on Dedicated Exadata Infrastructure for more details.
Ability to skip maintenance updates for a quarter.	You can customize your service maintenance preferences to skip patching a Release Update (RU) for one quarter. You cannot skip the maintenance updates for two consecutive quarters. See Update Autonomous Container Database Maintenance Preferences for more details.
Use DBMS_CLOUD_PIPELINE to create data pipelines for loading and exporting data in the cloud.	You can use DBMS_CLOUD_PIPELINE to create data pipelines for loading and exporting data in the cloud. This package supports continuous incremental data load of files in object store into the database, as well as continuous incremental export of table data or query results from the database to object store based on a timestamp column. See DBMS_CLOUD_PIPELINE Package for more information.
Use DBMS_CLOUD_REPO to use and manage cloud hosted code repositories from Oracle Database.	You can use DBMS_CLOUD_REPO to use and manage cloud hosted code repositories from Oracle Database. This package is a single interface for access to Multicloud Code repositories and allows you to upload SQL files to Git repositories or install SQL scripts directly from Cloud Code Repositories. This package also allows you to use a Cloud Code Repository to manage code versions for SQL scripts and to install or patch application code from Git repositories. Supported code repositories include GitHub, AWS CodeCommit, and Azure Repos. See About DBMS_CLOUD_REPO for more information.
Doc update: Use case demonstration to configure Autonomous AI Database on Dedicated Exadata Infrastructure with reference architecture.	Our documentation is updated with a use case demonstrating how to configure an Autonomous AI Database on Dedicated Exadata Infrastructure to leverage its capabilities better. See Use Case: Configure Autonomous AI Database with Reference Architecture for a comprehensive and recommended configuration that involves setting up separate development and production Autonomous AI Database environments.
Connect to Autonomous AI Database on Dedicated Exadata Infrastructure without a wallet.	You can connect your database applications or tools to an Autonomous AI Database on Dedicated Exadata Infrastructure without a wallet. Connecting an application without a wallet (TLS) provides security for authentication and encryption, and security is enforced using a security certificate that is trusted by the client operating system (OS). See Prepare for TLS Walletless Connections for more information.

Feature	Description
Bulk operations for files in the Cloud.	The PL/SQL package DBMS_CLOUD offers parallel execution support for bulk file upload, download, copy, and transfer activities, which streamlines the user experience and delivers optimal performance for bulk file operations. Refer to the following articles for more information: - Bulk Operations for Files in the Cloud - DBMS_CLOUD for Bulk File Management
Create a text index on the Object Storage files.	You can create a text index on the Object Storage files specified at the <code>location_uri</code> location with the DBMS_CLOUD. CREATE_EXTERNAL_TEXT_INDEX procedure. The index is refreshed at regular intervals, for any new additions or deletions done with files on location URI. See CREATE_EXTERNAL_TEXT_INDEX and DROP_EXTERNAL_TEXT_INDEX for reference.

March 2024

Feature	Description
Doc update: Overview of an Autonomous Container Database (ACD)	Our documentation has been updated with a new article to provide a complete overview of Autonomous Container Database resources. This article answers all questions about Autonomous Container Databases, including what an ACD is, the requirements for creating an ACD, which database features can be defined or managed using the ACD resource, and provides related documentation to help you create and manage ACDs. See About Autonomous Container Database.
Free developer database instances	Customers who have subscribed to Oracle Exadata Database Service or Autonomous AI Database on either Dedicated Exadata Infrastructure or Exadata Cloud@Customer can create and use Autonomous AI Database for Developers instances. Autonomous AI Database for Developers instances are free Autonomous AI Databases that developers can use to build and test new applications. With Autonomous AI Database for Developers instances, you can try new Autonomous AI Database features for free and apply them to ongoing or new development projects. Developer database comes with limited resources and is, therefore, not suitable for large-scale testing and production deployments. When you need more compute or storage resources, you can transition to a paid database licensing by cloning your developer database into a regular Autonomous AI Database. Refer to Autonomous AI Database for Developers for more details.

Feature	Description
Shared Server configuration for Autonomous AI Database applications	Autonomous AI Database on Dedicated Exadata Infrastructure supports Shared Server Configuration for maintaining legacy applications designed without connection pooling. You can enable shared server configuration while provisioning an Autonomous Container Database. For more details, see the Shared Server Configuration section in Special-Purpose Connection Features.
Autonomous AI Database resource management	Autonomous AI Database on Dedicated Exadata Infrastructure lets you configure the following attributes to a value that is most appropriate for your database while provisioning an Autonomous Container Database (ACD): - Database split threshold (CPU) - Node failover reservation (%) - Distribution affinity This capability improves your resource management ability to either consolidate more databases or have the highest database availability. See Create an Autonomous Container Database for more details.
Export data using DBMS_CLOUD.EXPORT_DATA	You can use the DBMS_CLOUD.EXPORT_DATA procedure to export data from an Autonomous AI Database on Dedicated Exadata Infrastructure to a directory or cloud Object Store. The export options are CSV, JSON, or XML. See the following pages for more details: - Export Data to a Directory Using DBMS_CLOUD.EXPORT_DATA - Export Data to Object Store as Text Using DBMS_CLOUD.EXPORT_DATA - EXPORT_DATA Procedure Reference - DBMS_CLOUD Package Format Options for EXPORT_DATA

February 2024

Feature	Description
Enable Autonomous Data Guard and add a standby database on an Autonomous Container Database on Exadata Cloud@Customer deployments.	On Exadata Cloud@Customer deployments, you can enable Autonomous Data Guard and add a standby Autonomous Container Database (ACD) to ACDs provisioned already. For more information, see Enable Autonomous Data Guard on an Autonomous Container Database.

Feature	Description
Use Select AI to generate SQL from Natural Language Prompts.	Autonomous AI Database can interact with AI service providers including: OpenAI and CohereAI to provide access to Generative AI capability from Oracle Database that uses Large Language Models (LLMs). One way LLMs can work with Oracle Database is by generating SQL from natural language prompts that allow you to talk to your database. See Use Select AI to Generate SQL from Natural Language Prompts for more information. To get a list of AI profiles and their attributes created in your Autonomous AI Database or schema, you can use the DBMS_CLOUD_AI views. See DBMS_CLOUD_AI Views for more information.
Send Microsoft Teams notifications from Autonomous AI Database on Dedicated Exadata Infrastructure on Exadata Cloud@Customer deployments.	On Exadata Cloud@Customer deployments, you have the capability to send messages, alerts, or the output of a query from Autonomous AI Database to a Microsoft Teams Channel. See Send Microsoft Teams Notifications from Autonomous AI Database for more information.
Send Slack notifications from Autonomous AI Database on Dedicated Exadata Infrastructure on Exadata Cloud@Customer deployments.	On Exadata Cloud@Customer deployments, you have the capability to send Slack notifications from Autonomous AI Database. This feature enables you to send messages to a Slack channel or share the results of a query to a Slack channel. See Send Slack Notifications from Autonomous AI Database for more information.
Rolling rotation of database SSL certificates.	With the rolling rotation of database SSL certificates, your existing database SSL certificate is not invalidated immediately after rotating it. <i>Within two weeks from the certificate rotation</i>, you can connect to your databases using the Autonomous AI Database client wallet you downloaded before or after the certificate rotation. <i>After two weeks from the certificate rotation:</i> • The database wallet <i>downloaded before the certificate rotation</i> is invalidated and can not be used to connect to your databases. • The database wallet <i>downloaded within two weeks</i> from the certificate rotation remains active and can be used to connect to your databases. • Any new database wallet <i>downloaded after two weeks</i> from the certificate rotation can be used to connect to your databases. Refer to Certificate Management for complete details.

Feature	Description
New events for Autonomous Container Databases.	You can get notified when your Autonomous Container Database's backup destination uses 80% of the storage space, runs out of storage space, has incorrect permissions set, or returns to a healthy state. To subscribe to event notifications, see Use Events and Notifications to Watch Resource Activity. Refer to Events for Autonomous AI Database on Dedicated Exadata Infrastructure for a complete list of events emitted by the Autonomous AI Database resources.

January 2024

Feature	Description
Send email using DBMS_CLOUD_NOTIFICATION from Autonomous AI Databases deployed on Oracle Public Cloud.	On Oracle Public Cloud, you can use the DBMS_CLOUD_NOTIFICATION procedures to send messages and query results as email. See Send Email on Autonomous AI Database on Dedicated Exadata Infrastructure for more information.
Send Microsoft Teams notifications from Autonomous AI Databases deployed on Oracle Public Cloud.	On Oracle Public Cloud, you have the capability to send messages, alerts, or query results from an Autonomous AI Database to a Microsoft Teams Channel. See Send Microsoft Teams Notifications from Autonomous AI Database for more information.
Send Slack notifications from Autonomous AI Databases deployed on Oracle Public Cloud.	On Oracle Public Cloud deployments, you have the capability to send Slack notifications from an Autonomous AI Database. This feature enables you to send messages or share the results of a query to a Slack channel. See Send Slack Notifications from Autonomous AI Database for more information.

New Feature Announcements (2023)

Announcements for the noteworthy changes made to Autonomous AI Database on Dedicated Exadata Infrastructure in 2023.

December 2023

Feature	Description
Disable and Enable Automatic Backups	By default, automatic backups are enabled for an Autonomous Container Database (ACD). However, you can now disable them while provisioning the ACD, and enable them anytime later. Once the automatic backups are enabled you can not disable them for an ACD. You can set the backup retention period for the backups between 7 to 95 days. Refer to the following for more information: - Create an Autonomous Container Database - Edit Autonomous Container Database Backup Settings
Ability to scale up or scale down an Autonomous Exadata VM Cluster	You can add resources to or remove them from an Autonomous Exadata VM Cluster (AVMC). This is called Autonomous Exadata VM Cluster vertical scaling. This ability lets you scale up or scale down an AVMC by modifying the following resources: - Number of CPUs per VM - Number of Autonomous Container Databases (ACDs) - Database storage For instructions, see Manage Autonomous Exadata VM Cluster Resources.

November 2023

Feature	Description
Doc Update: Hardened Security Configuration Features	We have added a new section to describe the strategies implemented by Autonomous AI Database to provide standard, hardened security configurations for your Autonomous VMs (Client VMs). Refer to Configuration Management in Autonomous AI Database for details.
Enable Autonomous Data Guard and add standby databases on an Autonomous Container Database.	On Oracle Public Cloud deployments, you can enable Autonomous Data Guard and add a standby Autonomous Container Database (ACD) to Autonomous Container Databases (ACD) provisioned already. For more information, see Enable Autonomous Data Guard on Autonomous Container Database.

Feature	Description
Oracle Real Application Testing (RAT): Database Replay	You can use Oracle Real Application Testing Database Replay to capture workload from an on-premises or other cloud service database and replay it on an Autonomous AI Database instance. This enables you to compare workloads between an on-premises database or other cloud service database and an Autonomous AI Database. You can also capture workload from an Autonomous AI Database on Dedicated Exadata Infrastructure and replay it on the same or any other Autonomous AI Database. This can be useful in scenarios where you want to test a patch or upgrade before applying it on your production database. See Using Oracle Real Application Testing for more information.

October 2023

Feature	Description
Documentation Update: Interactive graphic for Key Security Features	You can now refer to Security Features in Autonomous AI Database on Dedicated Exadata Infrastructure for complete information about the key security features supported by Autonomous AI Database on Dedicated Exadata Infrastructure. This document comes with a clickable graphic representing the building blocks for the key security features of the Autonomous AI Database. You can also click the block you are interested in exploring further.
DST Time-Zone file update for an Autonomous Container Database.	You can now choose to include or exclude a time-zone file update along with your quarterly automatic maintenance patches, either while provisioning an Autonomous Container Database (ACD) or for an existing ACD. Updating the time-zone file would require a complete downtime for the ACD and the associated Autonomous AI Databases. The downtime is dependent on the amount of data that is time-zone sensitive. You can also schedule an on-demand maintenance to update RU (Release Update) along with the time-zone file or just the time-zone file for an ACD. Refer to Schedule a Quarterly Maintenance Update for instructions.

Feature	Description
Rolling and Non-Rolling Maintenance methods to update an Autonomous Container Database.	You now have the option to choose between rolling or non-rolling maintenance methods to apply quarterly automatic maintenance patches. You can configure the maintenance method while provisioning an Autonomous Container Database (ACD) or for an existing ACD. The non-rolling maintenance method involves a full system downtime. See Service Maintenance Types for more information and Update Autonomous Container Database Maintenance Preferences for detailed instructions.
Documentation Update: Steps to configure CMU with Microsoft Active Directory on Exadata Cloud@Customer without using Oracle Object Store service.	Autonomous AI Database on Dedicated Exadata Infrastructure documentation is now updated with a new section that demonstrates how to configure CMU with Microsoft Active Directory on Exadata Cloud@Customer without using Oracle Object Store service. See Use Microsoft Active Directory with Autonomous AI Database on Dedicated Exadata Infrastructure for details.
New compute model: ECPU	You can configure your Autonomous AI Database resources using a new compute type called ECPUs. An ECPU is an abstracted measure of compute resources. ECPUs are based on the number of cores elastically allocated from a pool of compute and storage servers. For more information about the compute models supported by Autonomous AI Database, see Compute Models in Autonomous AI Database.
Enhanced support for resource usage tracking on Oracle Exadata Cloud@Customer	On Oracle Exadata Cloud@Customer deployments, you can monitor and track your Exadata resource allocation and usage across Autonomous Exadata VM Clusters (AVMC) and Autonomous Container Databases (ACD) with enhanced granularity and new visuals on the Oracle Cloud Infrastructure (OCI) console, in real-time. This release lets you get a comprehensive and clear picture of the compute and storage resources' allocation and usage at the AVMC and ACD levels from their Details page on the OCI console. Depending on your choice, you can view this information in a graphical or table view. Understanding and knowing these details can help you optimize your resource allocation to Autonomous AI Databases and efficiently forecast your capacity needs. For the exact list of the resource details you can monitor from the AVMC and ACD Details page and what they mean, refer to Plan and Monitor Capacity.

September 2023

Feature	Description
X10M systems available on Exadata Cloud@Customer deployments.	When you create Autonomous AI Database deployments on Exadata Cloud@Customer, you can now deploy to X10M Exadata Database Machine systems. For details about the capacities and characteristics of the X10M systems supported by Autonomous AI Database, see Exadata X10M Systems.
Limit Operator's Access to a Specific Customer-Approved Autonomous Container Database (ACD)	Autonomous AI Database on Dedicated Exadata Infrastructure lets you restrict Oracle operator access to a specific Autonomous Container Database (ACD) in an Autonomous Exadata VM Cluster (AVMC) in the diagnostic and maintenance cages. With this restriction enabled, you receive an approval request with the details to which the operators are requesting access. Once you review and approve the access request, the operators will get SQL access to only those approved ACDs. Refer to Auditing Oracle Operator Activities to know more about Oracle operator access control.
Parallel Replicat in Integrated Mode	Autonomous AI Database on Dedicated Exadata Infrastructure supports Parallel Replicat in integrated mode, that is, you can also create (or add) and configure parallel replication in your environment. See Using Parallel Replicat for more information.
 Note Parallel Replicat is not supported for overprovisioned Autonomous AI Databases, that is, databases provisioned with less than 1 OCPU.	
Documentation Update: Troubleshooting Guide	To help troubleshoot problems you might encounter with Oracle Autonomous AI Database on Dedicated Exadata Infrastructure, Troubleshooting Autonomous AI Database on Dedicated Exadata Infrastructure is added to our documentation.

August 2023

Feature	Description
Enhanced support for tracking resource usage on Oracle Public Cloud.	On Oracle Public Cloud deployments, you can monitor and track your Exadata resource allocation and usage across Autonomous Exadata VM Clusters (AVMC) and Autonomous Container Databases (ACD) with enhanced granularity and new visuals on the Oracle Cloud Infrastructure (OCI) console, in real-time. This release lets you get a comprehensive and clear picture of the compute and storage resources' allocation and usage at the AVMC and ACD levels from their Details page on the OCI console. Depending on your choice, you can view this information in a graphical or table view. Understanding and knowing these details can help you optimize your resource allocation to Autonomous AI Databases and efficiently forecast your capacity needs. For the exact list of the resource details you can monitor from the AVMC and ACD Details page and what they mean, refer to Plan and Monitor Capacity.
Database In-Memory Column Store support for Autonomous AI Database on Dedicated Exadata Infrastructure	You can enable Database In-memory for your Autonomous AI Database by allocating a percentage of its System Global Area (SGA) to the In-Memory column store (IM column store). You can enable or disable Database In-memory while or after provisioning the database with at least 4 OCPUs enabled. Scaling down the CPU count to less than 4 OCPUs later automatically disables it. You can also enable Database In-Memory on databases cloned from a database instance or backup, irrespective of whether the clone source has Database In-memory enabled. Please see Database In-Memory for the requirements and guidelines to use this feature with Autonomous AI Database. Refer to Enable or Disable Database In-Memory or Create an Autonomous AI Database on Dedicated Exadata Infrastructure for detailed instructions.

July 2023

Feature	Description
Cross region Autonomous Data Guard configuration for Autonomous AI Database on Exadata Cloud@Customer.	On Exadata Cloud@Customer deployments, you can specify an Exadata Infrastructure and Autonomous Exadata VM Cluster that is in a different region from the primary database's Exadata Infrastructure and Autonomous Exadata VM Cluster. This choice provides the highest level of protection against disasters, including a catastrophic loss of external network connectivity or power to an entire region. For more details about Autonomous Data Guard and related concepts, refer to Protect Critical Databases from Failures and Disasters Using Autonomous Data Guard.
Clone a database instance or its backup across regions and Exadata Infrastructure resources.	On Exadata Cloud@Customer deployments, you can: - Clone an Autonomous AI Database, with or without Autonomous Data Guard, to an Autonomous Container Database, with or without Autonomous Data Guard, on another Exadata Infrastructure in the same or a different region. In the case of Autonomous AI Databases with Autonomous Data Guard, you can clone only from the primary database. - Create a new Autonomous AI Database from a backup on the same or another Exadata Infrastructure in the same or a different region. In the case of Autonomous AI Databases that use Autonomous Data Guard, create a new database instance from either primary or standby database backups in the same or a different region. For step-by-step instructions, see: - Clone an Autonomous AI Database Instance - Clone Autonomous AI Database from Backup

June 2023

Feature	Description
Documentation update to consolidate the performance tuning and monitoring features supported by Autonomous AI Database on Dedicated Exadata Infrastructure.	The existing documentation related to the high-performance features supported by Autonomous AI Database on Dedicated Exadata Infrastructure is improvised to provide comprehensive information in a single article. The new article has: • An overview graphic to give you a bird's eye view of all the high-performance features and performance monitoring tools supported by the Autonomous AI Database. • A detailed explanation of each feature and tool with cross-references to the related how-to guides, wherever appropriate. Refer to High Performance Features in Autonomous AI Database on Dedicated Exadata Infrastructure for more information.
Base Shape support for Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Public Cloud.	Oracle Autonomous AI Database on Dedicated Exadata Infrastructure comes in different infrastructure shapes to support workloads of different sizes. In this release, the capability of Autonomous AI Database on Dedicated Exadata Infrastructure has been extended to support Base shape. See Resource Limits and Characteristics of Infrastructure Shapes for more information.
Use <code>DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION</code> to enable an external authentication scheme.	You can use <code>DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION</code> to enable Kerberos, Microsoft Azure AD, or Identity and Access Management (IAM) Authentication as an external authentication scheme for your Autonomous AI Database. Refer to the following topics for detailed instructions: • Enable Kerberos Authentication on Autonomous AI Database • Configure the Oracle Autonomous AI Database for Microsoft Azure AD Integration • Change External Identity Providers on Autonomous AI Database

Feature	Description
Use Directories to Load Data with DBMS_CLOUD Procedures.	As an alternative to an object store location URI, you can now specify a directory with DBMS_CLOUD procedures to load or unload data from files in a directory, including directories created on attached network file systems. The following procedures support specifying files in a directory with the <code>file_uri_list</code> parameter: - DBMS_CLOUD.COPY_COLLECTION - DBMS_CLOUD.COPY_DATA - DBMS_CLOUD.CREATE_EXTERNAL_TABLE The following procedures support specifying files in a directory with the <code>partitioning_clause</code> parameter: - DBMS_CLOUD.CREATE_EXTERNAL_TABLE - DBMS_CLOUD.CREATE_HYBRID_PART_TABLE Refer to DBMS_CLOUD for Objects and Files for more details.
Access Avro, ORC, and Parquet files in Object Stores.	You can now directly query and load data stored as Apache Avro, Apache Parquet, or ORC format files into Autonomous AI Database. You can also create external tables for Avro, Parquet or ORC format data files. Refer to the following topics for more details: - DBMS_CLOUD Avro, ORC, and Parquet Support - COPY_DATA Procedure for Avro, ORC, or Parquet Files - CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files

May 2023

Feature	Description
Compartment quota support for Autonomous AI Database on Exadata Cloud@Customer	Autonomous AI Database on Exadata Cloud@Customer now supports the compartment quotas feature of Oracle Cloud Infrastructure. Using this feature you can restrict, on a compartment by compartment basis, the availability of dedicated infrastructure resources in your tenancy. For more information, see Manage Resource Availability with Compartment Quotas. For general information about the compartment quotas feature, see Compartment Quotas.
Support for Autonomous VM Cluster node subsetting on Autonomous AI Database on Oracle Public Cloud.	On Oracle Public Cloud, you can now select the DB Servers while deploying a new Autonomous Exadata VM Cluster (AVMC) resource. This option is available only while provisioning an AVMC resource, and you can not change the selection after creating the AVMC successfully. For more details, refer to Create Autonomous Exadata VM Cluster.

Feature	Description
Monthly Infrastructure Security Maintenance Activity	Oracle schedules and performs monthly infrastructure security maintenance alongside quarterly maintenance. This maintenance activity is scheduled to be applied during a 21-day window that begins after the 15th of each month. You will receive notification of the proposed schedule at least 7 days before the start of the monthly maintenance window, and you can reschedule monthly maintenance to another date in the window if desired. These security patches are applied only in those months with critical security updates, including fixes for vulnerabilities with CVSS scores greater than or equal to 7. Please refer to Service Maintenance in Autonomous AI Database on Dedicated Exadata Infrastructure for more details.
 Note The Infrastructure Security Maintenance implementation will be rolled out to different Oracle Cloud Infrastructure (OCI) regions in a phased manner.	
Documentation Update: High Performance Features in Autonomous AI Database on Dedicated Exadata Infrastructure	The existing documentation explaining the High-Performance features supported by Autonomous AI Database on Dedicated Exadata Infrastructure is improvised and consolidated into a single article. This new article describes how to use the different performance monitoring and tuning features supported by Autonomous AI Database, referencing the related how-to guides. For more details, see High Performance Features in Autonomous AI Database.
Ability to bring your own certificates (BYOC)	On Oracle Public Cloud deployments, you can now seed your own security certificates with an Autonomous Exadata VM Cluster (AVMC) using the Oracle Cloud Infrastructure (OCI) console. This feature lets you use your CA-signed server-side Oracle REST Data Services (ORDS) or Secure Socket Layer (SSL) certificates while rotating the security certificates of your AVMC resource. Refer to Manage Security Certificates for an Autonomous Exadata VM Cluster Resource for more details.

Feature	Description
Ability to add storage to existing Exadata Infrastructure resource	You can now add additional storage servers to an existing Exadata Infrastructure resource and can create new Autonomous Exadata VM Clusters (AVMCs) using the newly added storage. In simple terms, you can scale your Exadata Infrastructure storage on-demand in an automated, elastic fashion. For more information on using this feature with <i>Oracle Public Cloud</i> deployments, see: • Scale CPU cores within an Exadata Cloud Infrastructure instance • Scale CPU cores in an Exadata Cloud Infrastructure cloud VM cluster or DB system • Add compute and storage resources to a flexible cloud Exadata infrastructure resource • Use the API to Manage Exadata Cloud Infrastructure Instance • cloud-exadata-infrastructures For more information on using this feature with <i>Exadata Cloud@Customer</i> deployments, see: • Use the Console to Scale Infrastructure Compute and Storage • Use the Console to Download Scale Infrastructure Compute and Storage Configuration File • Use the Console to Activate New Database and Storage Servers • Use the Console to Make Storage Capacity from New Server Available for VM Clusters Consumption • Use the Console to View Details of Exadata Cloud@Customer Infrastructure with Scaled Compute and Storage Capacity

April 2023

Feature	Description
Time Zone Handling in Calls to SYSDATE and SYSTIMESTAMP	The initialization parameter <code>SYSDATE_AT_DBTIMEZONE</code> parameter enables special handling in a session for the date and time value returned in calls to <code>SYSDATE</code> and <code>SYSTIMESTAMP</code>. Depending on the value of <code>SYSDATE_AT_DBTIMEZONE</code>, you see either the date and time based on the default Autonomous AI Database time zone, Coordinated Universal Time (UTC), or based on the time zone that you set in your database. See <code>SYSDATE_AT_DBTIMEZONE</code> for more information.

Feature	Description
SCAN listener port and TLS authentication mode	On Oracle Public Cloud deployments, you can now: - Choose a SCAN listener port for TLS and non-TLS from a range of available ports. - Choose between one-way TLS and mTLS for database TLS certificates. See Create an Autonomous Exadata VM Cluster for more details.
Access Network File System (NFS) or OCI File System Service (FSS) directories	You can now attach external NFS devices provided by the customer or OCI FSS using the <code>DBMS_CLOUD_ADMIN</code> procedures. This allows you to load data from external file storage in your Virtual Cloud Network (VCN) or any other Network File System in on-premises data centers. Refer to the following for more information: Attach External File Storage to Autonomous AI Database Attach Network File Storage to Autonomous AI Database DBMS_CLOUD_ADMIN Package
Base Shape support on Exadata Cloud@Customer deployments.	Oracle Autonomous AI Database on Dedicated Exadata Infrastructure comes in different infrastructure shapes to support workloads of different sizes. In this release, the capability of Oracle Exadata Cloud@Customer has been extended to support Base shape. See Resource Limits and Characteristics of Infrastructure Shapes for more information.
Bring your own certificates (BYOC)	You can now seed your own security certificates with an Autonomous Exadata VM Cluster (AVMC) on Exadata Cloud@Customer using the Oracle Cloud Infrastructure console. This feature lets you use your CA-signed server-side Oracle REST Data Services (ORDS) or Secure Socket Layer (SSL) certificates while rotating the security certificates of your AVMC resource. Refer to Manage Security Certificates for an Autonomous Exadata VM Cluster Resource for details.

March 2023

Feature	Description
Autonomous Container Database Name	While provisioning an Autonomous Container Database (ACD), you can now set the container database name of your choice from the Oracle Cloud Infrastructure (OCI) console. See Create an Autonomous Container Database for more details. In Exadata Cloud@Customer deployments, container database name is used to name the Oracle Key Vault's (OKV) wallet. You can use this name to locate the OKV wallet associated with your Autonomous Container Database on your OKV console.
Kerberos Authentication Support	You can now use Kerberos to authenticate your Autonomous AI Database users. For detailed guidance on configuring Kerberos authentication, its prerequisites and limitations, refer to Authenticate Autonomous AI Database Users With Kerberos.
SQL Tracing for Database Session	You can now use SQL tracing to help you identify the source of an excessive database workload, such as a high load SQL statement in your application. See Use SQL Tracing on Autonomous AI Database for more information.
Password Profile Support for User-Defined Profiles	You can now create and alter user profiles on Autonomous AI Database on Dedicated Exadata Infrastructure with a Password Verification Function (PVF). See Manage User Profiles with Autonomous AI Database for more information.
Oracle Client for Microsoft Tools (OCMT)	You can now download and use Oracle Client for Microsoft Tools (OCMT), which is a graphical user interface (GUI) native Microsoft Software installer (MSI) that simplifies ODP.NET setup and Oracle database connectivity to multiple Microsoft data tools. Refer to Connect Power BI and Microsoft Data Tools to Autonomous AI Database for more details and instructions on using OCMT.
Simplified Requirements: Clone From a Backup	We have now removed the requirement of one available Autonomous Container Database (ACD) and lowered the initial resource requirements to 1 OCPU for cloning an Autonomous AI Database from a backup. Refer to Clone Requirements for more details.
Long-Term Backups	You can now create long-term backups on Autonomous AI Database with a retention period anywhere from 90 days to 10 years. If needed, you can also update the retention period of a long-term backup to a different value after creating it. Refer to Backup and Restore Autonomous AI Databases for more details.

Feature	Description
Documentation Update: Backup and Restore Concepts	Our documentation is now reorganized to present all the concepts related to backup and recovery on a single page. Please refer to Backup and Restore Autonomous AI Databases for more details.

February 2023

Feature	Description
Updated documentation for DBMS_CLOUD package reference.	DBMS_CLOUD package reference is now updated with more procedures and functions. You can now refer to the following sections in Autonomous AI Database on Dedicated Exadata Infrastructure documentation for assistance with DBMS_CLOUD procedures and functions: • DBMS_CLOUD for Access Management • DBMS_CLOUD for Objects and Files • DBMS_CLOUD REST APIs
Support to save resource configuration as a stack	While provisioning new Autonomous AI Database resources from the Oracle Cloud Infrastructure console, you can now save the resource configuration as a stack with the click of a single button. You can then use the stack to create the resource through the Resource Manager service. A stack is a collection of Oracle Cloud Infrastructure resources corresponding to a Terraform configuration. Each stack resides in the compartment you specify in a single region. You can then deploy the resources on a given stack across multiple regions. For more information, see stack. For instructions to use this feature with different Autonomous AI Database resources, see: • Create an Exadata Infrastructure Resource • Create an Autonomous Exadata VM Cluster • Create an Autonomous Container Database • Create an Autonomous AI Database
Support for Autonomous VM Cluster node subsetting on Autonomous AI Database on Exadata Cloud@Customer	On Exadata Cloud@Customer deployments, you can now select the DB Servers while deploying a new Autonomous Exadata VM Cluster (AVMC) resource. This option is available only while provisioning an AVMC resource, and you can not change the selection after creating the AVMC successfully. For more details, refer to Create Autonomous Exadata VM Cluster.

Feature	Description
Support for snapshot standby database in an Autonomous Data Guard setup	You can now convert a standby database to a snapshot standby database by changing the standby Autonomous Container Database (ACD) to a snapshot standby ACD. See Convert Physical Standby to Snapshot Standby for step-by-step instructions. A snapshot standby database allows read-write operations on its data while maintaining minimum or zero data loss based on the protection mode. It receives and archives, but does not apply, redo data from the primary database. A snapshot standby database is helpful when you need to use your standby database for read-write operations, for instance, to test a patch before applying it to the primary database. Refer to Snapshot Standby Database for the requirements and limitations of the snapshot standby database.
Ability to update Protection mode and Fast start failover lag time	You can now change the protection mode of an Autonomous Data Guard setup. You can also modify the Fast start failover lag limit to any value between 5 to 3600 seconds for a primary ACD with automatic failover in an Autonomous Data Guard setup. See Update Autonomous Data Guard Settings for detailed instructions.
Oracle Home version control	You can now choose the Oracle Home version while provisioning an Autonomous Container Database on Dedicated Exadata Infrastructure. You have an option to choose between the latest Oracle Database version supported by Autonomous AI Database or its immediate predecessor. For example: Suppose the latest Oracle Database version supported by Autonomous AI Database is 19.18.0.1.0. Then, the Select base image drop-down on the Create Autonomous Container Database page lists 19.18.0.1.0 and 19.17.0.1.0 for you to choose. Refer to Create Autonomous Container Database for more details.
Ability to configure maintenance version of an Autonomous Container Database	You can now configure the maintenance type of an Autonomous Container Database to either Next RU or Latest RU to update to the next release update or the latest release update in the next maintenance window, respectively. Refer to Change the Maintenance Type of an Autonomous Container Database for more details.

New Feature Announcements

Announcements for the noteworthy changes made to Autonomous AI Database on Dedicated Exadata Infrastructure.

December 2022

Feature	Description
Documentation Update: Availability Service Level Agreements (SLAs) for Autonomous AI Database on Dedicated Exadata Infrastructure	Autonomous AI Database on Dedicated Exadata Infrastructure documentation is updated with the Service Level Agreements (SLAs) and Service Level Objectives (SLOs) for the Oracle Public Cloud and Oracle Exadata Cloud@Customer deployments. Refer to Availability Service Level Agreements (SLAs) for complete details.

Feature	Description
Multiple VM Autonomous AI Database on Dedicated Exadata Infrastructure	On Oracle Public Cloud deployments, you can now: • Create both Autonomous Exadata VM Clusters and Exadata Database VM Clusters on a common Exadata Infrastructure resource. • Create multiple Autonomous Exadata VM Clusters on an Exadata Infrastructure resource. • Support for multiple VM Clusters lets you: – Schedule separate maintenance runs for each Autonomous VM Cluster on the same Exadata Infrastructure. – Choose different license models for Autonomous AI Databases on the same Exadata Infrastructure. – Create and test Autonomous Data Guard between Autonomous Exadata VM Clusters on the same Exadata Infrastructure. – Customize compute, storage, and memory of each Autonomous Exadata VM Cluster configuration for the intended workload. – Implement network isolation of Autonomous Exadata VM Clusters to provide security and performance guarantees for specific workloads (Dev-Test, Staging, and Production).

Note

You can only create a Multiple VM - Autonomous VM Cluster (AVMC) on Exadata Infrastructure resources created in Oracle Cloud after the launch of the multiple VM Autonomous AI Databases feature. Please create a service request in My Oracle Support if you need to address this limitation and want to add an Autonomous VM Cluster to older Exadata Infrastructure resources. For help with filing a support request, see [Create a Service Request in My Oracle Support](#).

Feature	Description
Documentation Update: IAM Policy Details for Autonomous AI Database on Dedicated Exadata Infrastructure	Autonomous AI Database on Dedicated Exadata Infrastructure documentation is updated with the details for writing policies to control access to Autonomous AI Database resources. Refer to Policy Details for Oracle Autonomous AI Database for complete details.

November 2022

Feature	Description
SCAN listener port and TLS authentication mode	On Exadata Cloud@Customer deployments, you can now choose: - A SCAN listener port for TLS and non-TLS from a range of available ports. - Between one-way TLS and mTLS for database TLS certificates. The above choices are available <i>only</i> while provisioning an Autonomous Exadata VM Cluster (AVMC) resource. Refer to Create an Autonomous Exadata VM Cluster for detailed instructions.
Connect Python applications using the python-oracledb driver.	On Exadata Cloud@Customer deployments, you can now connect Python applications to your database instance using the python-oracledb driver. Refer to Connect Python Applications Using mTLS for more details.

October 2022

Feature	Description
Oracle LogMiner	You can now use LogMiner with your Autonomous AI Database. LogMiner, which is part of Oracle Database, enables you to query online and archived redo log files through a SQL interface. For limitations on using LogMiner with Autonomous AI Database, see Limitations on the Use of LogMiner.
Oracle Database API for MongoDB	Using MongoDB commands, you can now use Oracle Database API for MongoDB to build applications that interact with collections of JSON documents in an Autonomous AI Database. Refer to Oracle API for MongoDB Support for the requirements and procedure involved in using Oracle Database API for MongoDB with an Autonomous AI Database.

August 2022

Feature	Description
New events for Autonomous Data Guard associations.	You can now get notified when an Autonomous AI Database changes its role from standby to primary or primary to standby as a result of manual failover, switchover, or automatic failover. To subscribe to event notifications, see Use Events and Notifications to Watch Resource Activity. Refer to Events for Autonomous AI Database on Dedicated Exadata Infrastructure for a complete list of events emitted by the Autonomous AI Database resources.
Create a new Autonomous AI Database instance from backup.	You can now create a new Autonomous AI Database by cloning from a backup on the same or another Exadata Infrastructure in the same or a different region. In the case of dedicated databases that use Autonomous Data Guard, you can create a new database instance using the backup of either primary or standby databases. For more details, see: • Clone Sources • Clone Autonomous AI Database from Backup
Clone a database instance across regions and Exadata Infrastructure resources.	You can now clone an Autonomous AI Database to an Autonomous Container Database on another Exadata Infrastructure in the same or a different region. For step-by-step instructions, see Clone an Autonomous AI Database Instance.
Fast Ingest Support	You can now optimize the processing of high-frequency, single-row data inserts into Autonomous AI Database using fast ingest. To learn more about using fast ingest with Autonomous AI Database, see Limitations on the Use of Fast Ingest.

July 2022

Feature	Description
Oracle Operator Access Control for Autonomous AI Database on dedicated exadata infrastructure.	Operator Access Control is now expanded to provide controls for client virtual machines deployed on Oracle Autonomous AI Database on Dedicated Exadata Infrastructure on dedicated exadata infrastructure. Similar to Operator access control for Exadata Cloud@Customer Infrastructure, customers may now impose Oracle operator access controls on their Autonomous Virtual Machine clusters deployed on Oracle cloud. See Auditing Oracle Operator Activities for more details.

Feature	Description
Documentation addition about Ops Insights.	A new article is added to the Autonomous AI Database documentation that describes how you can use Ops Insights to monitor databases. Refer to Use Ops Insights on Autonomous AI Database on Dedicated Exadata Infrastructure for more details.
Badging the features exclusive to either Oracle cloud or Exadata Cloud@Customer deployments.	Articles discussing the features that are exclusive to either Oracle cloud or Exadata Cloud@Customer deployments have been updated with the following badges, as appropriate: • APPLIES TO:  Oracle Public Cloud only • APPLIES TO:  Exadata Cloud@Customer only For a glance at the features supported exclusively by Oracle cloud or Exadata Cloud@Customer deployment choices, refer to Feature Parity Between Oracle Cloud and Exadata Cloud at Customer.

June 2022

Feature	Description
Azure Active Directory (Azure AD) Integration	Azure Active Directory (Azure AD) users can now connect to an Autonomous AI Database instance using Azure OAuth2 access tokens. See Authenticate and Authorize Microsoft Azure Active Directory Users for Autonomous AI Database for more information.
New events for Cloud Autonomous Exadata VM Cluster and Autonomous AI Database resources.	You can now get notified when: • An SSL certificate is rotated either manually (using the Oracle Cloud Infrastructure console or API) or automatically on its expiry. • Your Autonomous AI Database wallet has been rotated and must be downloaded. To subscribe to event notifications, see Use Events and Notifications to Watch Resource Activity . Refer to Events for Autonomous AI Database on Dedicated Exadata Infrastructure for a complete list of events emitted by Autonomous AI Database resources.
Time zone selection for Autonomous Exadata VM Clusters on Oracle Cloud.	You can now set a time zone other than UTC or the browser-detected time zone while provisioning an Autonomous Exadata VM Cluster on Oracle Cloud. For more details, see Create an Autonomous Exadata VM Cluster.

May 2022

Feature	Description
Elastic provisioning for X8M-2 and X9M-2 systems on Oracle Cloud.	While provisioning X8M-2 and X9M-2 systems on Oracle Cloud, you can now choose a server configuration that is different from the default configuration. See Create an Exadata Infrastructure Resource for specific details.
Documentation addition about monitoring databases with Autonomous AI Database metrics.	A new article is added to the Autonomous AI Database documentation that describes how you can use Autonomous AI Database metrics to monitor databases. The complete list of available metrics for the Autonomous AI Database is also documented for reference. See Monitor Databases with Autonomous AI Database Metrics for more details.
Character set selection.	You can now choose a character set of your choice from a list of supported character sets while provisioning an Autonomous AI Database. Even though you are allowed to choose any supported combination of database and national character sets for an Autonomous AI Database, the Autonomous Container Database continues to use AL32UTF8 as the database character set and AL16UTF16 as the national character set. Oracle supports different database and national character sets inside an Autonomous Container Database for each database. See Character Set Selection for Dedicated Autonomous AI Database for more details.
Oracle standard tagging for Autonomous AI Database cloud resources on Exadata Cloud@Customer.	You can now use Oracle Standard tags to tag your Cloud@Customer cloud resources as per your organizational scheme. Tagging allows you to group resources, manage costs, and give insights into usage. For more details on tagging your cloud resources, see Tagging Dedicated Autonomous AI Database Cloud Resources.

Feature	Description
Support for resource usage tracking.	You can now monitor and manage your resource consumption easily with the help of OCPU and storage details displayed on the Oracle Cloud Infrastructure console for your Autonomous Exadata VM Cluster (AVMC) and Autonomous Container Database (ACD) resources. As part of this feature release, the following details are displayed on the Oracle Cloud Infrastructure console for your Autonomous Exadata VM Cluster and Autonomous Container Database resources: • Total OCPUs • Available OCPUs • Reclaimable OCPUs • Available Storage in TB • Total Storage in TB • Memory per OCPU • Total number of ACDs that you can provision in this resource in an AVMC resource. • Remaining (available) ACDs that you can provision in an AVMC resource. Understanding and knowing these details can help you effectively track and manage the OCPU and storage resources in your Autonomous Exadata VM Clusters and Autonomous Container Databases. See OCPU Management and Billing for more details.
Increase in Database Name length from 14 characters to 30 characters.	You can now enter a value of up to 30 characters for Database Name while creating or cloning an Autonomous AI Database. Review the following guidelines when selecting a database name while creating or cloning an Autonomous AI Database: • Maximum 30 characters in length • Contain only alphanumeric characters • Begin with an alphabetic character • Not contain spaces

April 2022

Feature	Description
Oracle standard tagging for Autonomous AI Database cloud resources	You can now use Oracle Standard tags to tag your cloud resources as per your organizational scheme. Tagging allows you to group resources, manage costs, and give insights into usage. For more details on tagging your cloud resources, see Tagging Dedicated Autonomous AI Database Cloud Resources.

Feature	Description
Documentation addition listing API to manage the Autonomous Data Guard configuration.	A reference article is added to the Autonomous AI Database documentation to provide a list of REST API endpoints to manage Autonomous Data Guard configuration. See API to Manage Autonomous Data Guard Configuration
Documentation update consolidating the service maintenance concepts and related how-to guides.	The existing documentation explaining the concepts related to Service Maintenance of an Autonomous AI Database is improvised and consolidated into a single article. This new article references all the related how-to guides that explain how to view and manage service maintenance for the different infrastructure resources. For more details, see Service Maintenance of Dedicated Autonomous AI Database.
Documentation addition listing the IP requirements to create an Autonomous Exadata VM Cluster on Oracle Cloud.	A reference article is added to the Autonomous AI Database documentation to list the minimum required subnet sizes for Autonomous AI Database deployments on Oracle Cloud to create an Autonomous Exadata VM Cluster successfully. For more details, see IP Requirements for Autonomous Exadata VM Cluster.
X9M-2 systems available for Autonomous AI Database on Oracle Public Cloud deployments.	When you create Autonomous AI Database deployments on Oracle Public Cloud, you can now deploy to X9M-2 Exadata Database Machine systems. For details about the capacities and characteristics of the X9M-2 systems supported by the Autonomous AI Database, see Exadata X9M-2 Systems.
Create a new Autonomous AI Database instance from backup.	You can now create a new Autonomous AI Database by cloning from a backup on Exadata Cloud@Customer. In the case of dedicated databases that use Autonomous Data Guard, you can create a new database instance using the backup of either <i>primary</i> or <i>standby</i> databases. For more details, see Clone an Autonomous AI Database.
New event for Cloud Autonomous Exadata VM Cluster resources.	You can now get a reminder to rotate an SSL certificate when it is older than 365 days. To subscribe to event notifications, see Use Events and Notifications to Watch Resource Activity. For a complete list of events emitted by Cloud Autonomous Exadata VM Cluster resources, see Cloud Autonomous Exadata VM Cluster Event Types.

March 2022

Feature	Description
Identity and Access Management (IAM) Integration with Autonomous AI Database	With the latest Release Update (RU Version 19.14.0.1), you can now configure Autonomous AI Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous AI Database with IAM credentials. As of this release, (IAM) authentication and authorization: • Will be enabled on all the newly provisioned Autonomous Container Databases (ACDs) and Autonomous AI Databases by default. • Can not be used with Autonomous AI Databases configured with Autonomous Data Guard. See Use Identity and Access Management (IAM) Authentication with Autonomous AI Database for more details.
Direct Data Pump Export to Cloud Object Store	With the Release Update (RU Version 19.13.0.1), you can now export data from Autonomous AI Database to Cloud Object Store directly. See Export Data From Autonomous AI Database to Object Store Directly for more details.
Multiple VM Autonomous AI Database	On Exadata Cloud@Customer deployments, you can now: • Create both Autonomous Exadata VM Clusters and Exadata Database VM Clusters on a common Exadata Infrastructure resource. • Create multiple Autonomous Exadata VM Clusters on a single Exadata Infrastructure resource. • Support for multiple VM Clusters lets you: – Schedule separate maintenance runs for each Autonomous VM Cluster on the same Exadata Infrastructure. – Choose different license models for Autonomous AI Databases on the same Exadata Infrastructure. – Create and test Autonomous Data Guard between Autonomous Exadata VM Clusters on the same Exadata Infrastructure. – Customize compute, storage, and memory of each Autonomous Exadata VM Cluster configuration for the intended workload. – Implement network isolation of Autonomous Exadata VM Clusters to provide security and performance guarantees for specific workloads (Dev-Test, Staging, and Production). See Create an Autonomous Exadata VM Cluster for more details.

Feature	Description
Elastic provisioning for X8M-2 systems on Oracle Cloud	While provisioning an Exadata Infrastructure resource on Oracle Cloud, you can now choose a server configuration that is different from the default configuration. See Create an Exadata Infrastructure Resource for specific details.

February 2022

Feature	Description
Blockchain Tables Support	Blockchain tables are insert-only tables that organize rows into a number of chains. Blockchain tables protect data that records important actions, assets, entities, and documents from unauthorized modification or deletion by criminals, hackers, and fraud. Blockchain tables prevent unauthorized changes made using the database and detect unauthorized changes that bypass the database. See Managing Blockchain Tables for more information.
Immutable Tables Support	Immutable tables are read-only tables that prevent unauthorized data modifications by insiders and accidental data modifications resulting from human errors. See Managing Immutable Tables for more information.
New events for Cloud Autonomous Exadata VM Cluster and Autonomous AI Database resources.	You can now get notified of new events emitted by the Autonomous Exadata VM Cluster, Autonomous Container Database, and Autonomous AI Database infrastructure resources. Refer to the following topics for a complete list of events emitted by these infrastructure resources: - Cloud Autonomous Exadata VM Cluster Event Types - Autonomous Container Database Event Types - Autonomous AI Database Event Types

January 2022

Feature	Description
Documentation addition to list the event types supported by Autonomous AI Database on Dedicated Exadata Infrastructure.	A reference article is added to Autonomous AI Database documentation to provide a list of different event types supported by Autonomous AI Database on Oracle Cloud. Using the Events service, you can create rules to capture these events and perform actions such as sending you emails using the Notifications service. See Event Types for Autonomous AI Database on Dedicated Exadata Infrastructure for more information.

Feature	Description
Payment Card Industry Data Security Standard (PCI DSS) compliance	Autonomous AI Database on Dedicated Exadata Infrastructure is now compliant with Payment Card Industry Data Security Standard (PCI DSS). Payment Card Industry Data Security Standard is a set of requirements intended to ensure that all companies that process, store, or transmit credit card information maintain a secure environment. See Regulatory Compliance Certification to view the complete list of international and industry-specific compliance standards met by Autonomous AI Database on Dedicated Exadata Infrastructure.
Automatic Failover or Fast-Start Failover	Autonomous AI Database deployments with Autonomous Data Guard can now failover to their standby Autonomous Container Databases automatically whenever the primary Autonomous Container Database becomes unavailable. This automatic failover is also known as Fast-Start Failover. For more details about fast-start failover, see Autonomous Data Guard to Protect Critical Databases from Failures and Disasters.

New Feature Announcements (2021)

Announcements for the noteworthy changes made to Autonomous AI Database on Dedicated Exadata Infrastructure in the year 2021.

December 2021

Feature	Description
Introduction of Autonomous Exadata VM Cluster and shift to a common Exadata Infrastructure resource for all Exadata Cloud-based services, which includes Autonomous AI Database on Dedicated Exadata Infrastructure.	The infrastructure setup of a Autonomous AI Database is taking a new shape with the introduction of a new Autonomous Exadata VM Cluster infrastructure resource. The primary changes are as follows: - Exadata Infrastructure (EXA) replaces the Autonomous Exadata Infrastructure resource in the existing model. - Autonomous Exadata VM Cluster (AVM) that you can now provision on top of the Exadata Infrastructure resource. This is a new layer on which you can now define Autonomous Container Databases. - Networking, License configuration and Certificate Management shifts from the Exadata Infrastructure resource to the Autonomous Exadata VM Cluster resource. - New events on the Exadata Infrastructure and Autonomous VM Cluster resources. Further Reference: - For more details, see Overview of Latest Resource Model. - For a comprehensive comparison with the current resource model, see Comparison with Current Resource Model. - To understand how your existing Autonomous AI Database infrastructure resources update to the latest resource model, see Updating to Latest Resource Model.
Encryption Key Version OCID and Key Rotation History	You can now view the Key Version OCID every time you rotate a customer-managed key for your Autonomous Container Database. The details page for the corresponding Autonomous Container Database and its associated Autonomous AI Databases will now show the latest Key Version OCID and Key History.

November 2021

Feature	Description
Oracle Operator Access Control for Autonomous AI Database on Oracle Exadata Cloud@Customer deployments.	Operator Access Control is now expanded to provide controls for client virtual machines deployed on Autonomous AI Database on Dedicated Exadata Infrastructure on Exadata Cloud@Customer. Similar to Operator access control for Exadata Cloud@Customer Infrastructure, customers may now impose Oracle operator access controls on their Autonomous Virtual Machine clusters deployed on Exadata Cloud@Customer. See Auditing Oracle Operator Activities for more details.
Documentation addition to describe sending mail with UTL_SMTP.	The documentation is updated to provide the steps and sample code for sending email using UTL_SMTP. See Sending Mail on Dedicated Autonomous AI Database for more information.
Integration with Database Management Service.	You can now use Database Management Service to monitor the health of a single Autonomous AI Database or a fleet of Autonomous AI Databases. For more details about the Database Management Service and learning how to enable it for your Autonomous AI Database, see Use Database Management Service to Monitor Databases.

October 2021

Feature	Description
UTL_HTTP and UTL_SMTP Database PL/SQL Packages.	You can now use UTL_HTTP and UTL_SMTP packages from your Autonomous AI Database on Dedicated Exadata Infrastructure. See Notes on the Use of Database PL/SQL Packages for more details.
Customer-managed encryption keys for Autonomous Container Databases with cross region deployments.	You can now use customer-managed encryption keys with Autonomous Data Guard-enabled Autonomous Container Databases with the primary and standby databases located in different regions. For information about provisioning an Autonomous Container Database with cross-region Dataguard using your own keys, see Create an Autonomous Container Database. For information on using customer-managed keys in Autonomous AI Database on Dedicated Exadata Infrastructure, see the following topics: • Prepare to Use Customer-Managed Keys • Use Customer-Managed Keys

September 2021

Feature	Description
X9M-2 systems available on Exadata Cloud@Customer deployments.	When you create Autonomous AI Database deployments on Oracle Exadata Cloud@Customer, you can now deploy to X9M-2 Exadata Database Machine systems. For details about the capacities and characteristics of the X9M-2 systems supported by Autonomous AI Database, see Exadata X9M-2 Systems .
New tile for SODA on the Details page of Autonomous AI Database.	You can now download SODA drivers for an Autonomous AI Database from its Details page. See Simple Oracle Document Access (SODA) for detailed instructions.

July 2021

Feature	Description
Documentation addition to explain how to enable and configure <i>Shared Server Configuration</i> for Autonomous AI Database applications.	Autonomous AI Database on Dedicated Exadata Infrastructure supports Shared Server Configuration for maintaining legacy applications designed without connection pooling. Autonomous AI Database documentation is updated to provide information about the Shared Server Configuration and explain how to enable it for Autonomous AI Database applications. See the Shared Server Configuration section in <i>Connect for High Performance</i> .

June 2021

Feature	Description
Fractional OCPUs and GB Storage.	You can now create or clone Autonomous AI Databases with less than 1 OCPU count using fractional units from 0.1 to 0.9 OCPUs and GB sizing between 32 GB and the maximum usable storage for your Exadata shape. For more information, see: - Create a Dedicated Autonomous AI Database - Clone a Dedicated Autonomous AI Database You can also scale up from or scale down to a fractional OCPU from the Details page of your Autonomous AI Database. For more information, see: - Add CPU or Storage Resources to a Dedicated Autonomous AI Database - Remove CPU or Storage Resources from a Dedicated Autonomous AI Database

Feature	Description
Customer-managed keys in Oracle Key Vault (OKV) for Autonomous AI Database enabled with Autonomous Data Guard on Exadata Cloud@Customer	You can now integrate your on-premises Oracle Key Vault (OKV) with Autonomous Data Guard enabled Autonomous AI Databases on Exadata Cloud@Customer and use customer-managed keys stored in Oracle Key Vault to secure your critical data. See <i>Prepare to Use Customer-Managed Keys in the Vault Service</i> for more details.

May 2021

Feature	Description
Restrict Access Using a Network Access Control List	You can now define additional security for your Autonomous AI Database on Dedicated Exadata Infrastructure deployments by setting up network Access Control Lists (ACLs) with specific IP addresses or CIDR blocks. This feature which was earlier supported only on Autonomous AI Database on Dedicated Exadata Infrastructure deployment on Oracle Exadata Cloud@Customer is now extended to Autonomous AI Database on Dedicated Exadata Infrastructure deployments on <i>Oracle Cloud</i> . An ACL allows you to formulate a fine-grained access control policy by limiting your Autonomous AI Database's network access to specific applications or clients. See <i>Set Access Control List for Dedicated Autonomous AI Database</i> .
Oracle Operator Access Control for Autonomous AI Database on Oracle Exadata Cloud@Customer deployments.	Oracle Operator Access Control is a compliance audit system that enables you to maintain close management and audit trails of all actions that an Oracle operator performs on the infrastructure. You can now use Oracle Operator Access Control in your Autonomous AI Database on Oracle Exadata Cloud@Customer deployments. For more information, see <i>Auditing Oracle Operator Activities in Autonomous AI Database on Dedicated Exadata Infrastructure Documentation</i> .

April 2021

Feature	Description
Customer-managed encryption keys for Autonomous Container Databases with cross availability domain Autonomous Data Guard deployments.	You can now use customer-managed encryption keys with Autonomous Data Guard-enabled Autonomous Container Databases with the primary and standby databases located in different availability domains within the same region.

Feature	Description
Support for Oracle GoldenGate Capture and Replicat	You can now replicate data to an Autonomous AI Database using Oracle GoldenGate and Oracle GoldenGate Cloud Service. You can capture changes from the Autonomous AI Database and replicate to any target database or platform that Oracle GoldenGate supports, including other Autonomous AI Database. See Use Oracle GoldenGate to Replicate Data with Autonomous AI Database.
Oracle Cloud Infrastructure navigation menu changes.	The Oracle Cloud Infrastructure console left navigation menu is upgraded. Some of the improvements include: - Options to customize the menu by pinning your favorite services for quick access or simply picking from your Most Recently Used ones. - Improved categorization of services into logical groups, combined with an inline search of menu items, makes it easier to find services. - Overview pages for top-level categories and individual services provide a top-level introduction or summary view of the category or service.
X8M systems available for Autonomous AI Database on Dedicated Exadata Infrastructure deployments on Public Cloud.	When you create dedicated Autonomous AI Database deployments on Oracle Public Cloud, you can now deploy to X8M Exadata Database Machine systems. For details about the capacities and characteristics of the X8M systems supported by Autonomous AI Database, see Characteristics of Infrastructure Shapes.
Support for software protected keys.	You can now provision Autonomous Container Databases using master encryption key that is stored in a software file system in the Vault service. For details about using customer -managed keys in Oracle Key Vault, see Prepare to Use Customer-Managed Keys in the Vault Service.

February 2021

Feature	Description
Oracle Database Actions	SQL Developer Web is now called Database Actions. Database Actions is a web-based interface that provides development tools, data tools, administration, and monitoring features for Autonomous AI Database. Using Database Actions, you can load data and run SQL statements, queries, and scripts in a worksheet. See Connect with Database Actions for more information.

Feature	Description
One-off patching for critical service requests	Oracle can now generate one-off patches for critical service requests filed with My Oracle Support. When you and Oracle agree that a service request is critical and requires a one-off patch for immediate resolution, the service team generates a one-off patch and makes it available. To learn how to manage one-off patches, see Service Maintenance of Dedicated Autonomous AI Database in <i>Autonomous AI Database on Dedicated Exadata Infrastructure Documentation</i> .

January 2021

Feature	Description
Restrict Access using a network access control list for Autonomous Data Guard enabled dedicated databases.	You can now define separate access control lists (ACLs) for the primary and standby database in Autonomous Data Guard enabled dedicated databases on Oracle Exadata Cloud@Customer. By default, the standby database inherits the ACLs defined for the primary database. However, you can specify a different ACL for the standby database, if needed. For more details, see Set Access Control List for Dedicated Autonomous AI Database in <i>Autonomous AI Database on Dedicated Exadata Infrastructure Documentation</i> .

New Feature Announcements (2020)

Announcements for the noteworthy changes made to Autonomous AI Database on Dedicated Exadata Infrastructure in the year 2020.

Topics:

December 2020

Feature	Description
Oracle Application Express (APEX) 20.2 is now available for new Autonomous Container Databases.	You can now use Oracle Application Express 20.2 to develop applications for an Autonomous AI Database. Only newly created Autonomous Container Databases have access to Oracle Application Express 20.2. All the existing Autonomous Container Databases will continue with Oracle Application Express 19.1. See Creating Applications with Oracle Application Express on Autonomous AI Database.
 Note To use Oracle Application Express 20.2 features for enhancing the existing applications, you must create a new Autonomous Container Database and migrate your data and application to it from the existing Autonomous Container Database.	
Rotate the Oracle-managed self-signed certificates from the Oracle Cloud Infrastructure console to meet your organization's security compliance needs.	You can now rotate the security certificates for Oracle REST Data Services (ORDS) and Secure Socket Layer (SSL) from the details page of an Autonomous Exadata Infrastructure. See Manage Security Certificates in <i>Autonomous AI Database on Dedicated Exadata Infrastructure Documentation</i>.

November 2020

Feature	Description
Restrict Access Using a Network Access Control List	You can now define additional security to your Autonomous AI Database on Dedicated Exadata Infrastructure deployments on Oracle Exadata Cloud@Customer by setting up network Access Control Lists (ACLs) with specific IP addresses or CIDR blocks. An ACL allows you to formulate a fine-grained access control policy by limiting your Autonomous AI Database's network access to specific applications or clients. See Set Access Control List for Dedicated Autonomous AI Database.

October 2020

Feature	Description
Use Autonomous Data Guard to protect against disasters and failures	You can now create Autonomous Container Databases with Autonomous Data Guard enabled. This feature enables you to keep your critical production databases available to mission critical applications despite failures, disasters, human error, or data corruption. To provide this capability, it creates and maintains two completely separate copies of your database: a <i>primary</i> database that your applications connect to and use, and a <i>standby</i> database that is a synchronized copy of the primary database. Then, should the primary database become unavailable for any reason, Autonomous Data Guard automatically converts the standby database to the primary database and, as such, it begins servicing your applications. See Configure and Manage Autonomous Data Guard for more information.
Manage your own encryption keys	Normally, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure creates and manages the encryption keys used to protect your data. Now, Oracle Autonomous AI Database on Dedicated Exadata Infrastructure can instead use keys you create and manage in the Oracle Cloud Infrastructure Vault service or Oracle Key Vault, depending on whether you are deploying Oracle Autonomous AI Database on Oracle Cloud or on Exadata Cloud@Customer. See Manage Master Encryption Keys for more information.

September 2020

Feature	Description
X8M-2 systems available on Exadata Cloud@Customer deployments	When you create dedicated Autonomous AI Database deployments on Oracle Exadata Cloud@Customer you can now deploy to X8M-2 Exadata Database Machine systems. For details about the capacities and characteristics of the X8M-2 systems supported by Autonomous AI Database, see Characteristics of Infrastructure Shapes.

August 2020

Feature	Description
In-database support for Microsoft Active Directory users	You can configure an Autonomous AI Database to authenticate and authorize Microsoft Active Directory users. This allows Active Directory users to access the database using their Active Directory credentials. See Use Microsoft Active Directory with Autonomous AI Database for more information.

July 2020

Feature	Description
Manage upcoming maintenance events	You can now manage the actual timing of scheduled maintenance events on Autonomous Exadata Infrastructure and Autonomous Container Database resources, even to the point of performing the activity immediately or of skipping it altogether. For more information, see: • View and Manage Scheduled Maintenance of an Autonomous Exadata Infrastructure Resource • View and Manage Scheduled Maintenance of an Autonomous Container Database
Assess security of your database using Data Safe	You can now use Oracle Data Safe to identify and protect sensitive and regulated data in Autonomous AI Databases. Oracle Data Safe helps you understand the sensitivity of your data, evaluate risks to data, mask sensitive data, implement and monitor security controls, assess user security, monitor user activity, and address data security compliance requirements. For details about using Data Safe with your Autonomous AI Databases, see Use Data Safe to Assess and Protect Data.

June 2020

Feature	Description
Autonomous AI Database now available on Exadata Cloud@Customer	You can now create dedicated Autonomous AI Database deployments on Oracle Exadata Cloud@Customer in your data center. For more information, see Introduction to Autonomous AI Database on Exadata Cloud@Customer.
New Performance Hub features	Performance Hub provides these new features for Autonomous AI Databases: • A Blocking Sessions tab you can use to view and, if necessary, kill sessions that are waiting or are blocked by sessions that are waiting. • The ability to generate and download an AWR (Automatic Workload Repository) report that shows data captured between two points in time (called snapshots). • The ability to change the time zone that Performance Hub uses when selecting timespans and displaying data. For more information, see Use Performance Hub to Monitor Databases.

May 2020

Feature	Description
Full rack Exadata Database Machines now supported	Previously, you could create Autonomous Exadata Infrastructure resources only on quarter-rack or half-rack Exadata Database Machine systems. Now, you can create Autonomous Exadata Infrastructure resources on full rack Exadata Database Machine systems as well. For details about the capacities and characteristics of these systems, see Characteristics of Autonomous Exadata Infrastructure Resources.

April 2020

Feature	Description
Oracle Extensions for IDEs support Autonomous AI Databases	You can now use these IDE extensions to connect to, browse, and manage Autonomous AI Databases directly from within the IDE: • Oracle Cloud Infrastructure Toolkit for Eclipse • Oracle Developer Tools for Visual Studio • Oracle Developer Tools for VS Code For more information, see Use Oracle Extensions for IDEs to Develop Applications.

March 2020

Feature	Description
Auto scaling of databases now available	You can now enable auto scaling of Autonomous AI Databases. When auto scaling is enabled, the database can automatically use additional CPU and IO resources without any manual intervention if the database workload requires, up to three times the OCPU count assigned to the database. You can enable auto scaling when you create a database, as described in Create a Dedicated Database, or after it has been created, as described in Enable or Disable Auto Scaling of a Dedicated Database.
Clone a database to a different Autonomous Container Database	When you create a new Autonomous AI Database by cloning an existing one, you can now choose to create the cloned database in a different Autonomous Container Database from the source database. For instructions, see Clone a Dedicated Autonomous AI Database.
Monitor your database using Autonomous AI Database metrics	You can monitor the health, capacity, and performance of your databases with metrics, alarms, and, notifications. You can use the Oracle Cloud Infrastructure console or Monitoring APIs to view metrics. For more information, see Use Autonomous AI Database Metrics to Monitor Databases.

February 2020

Feature	Description
More Oracle Exadata Database Machine options	Previously, you could create Autonomous Exadata Infrastructure resources only on quarter-rack Exadata Database Machine X7-2 systems. Now, you can create Autonomous Exadata Infrastructure resources on quarter-rack or half-rack Exadata Database Machine X7-2 or X8-2 systems. For details about the capacities and characteristics of these systems, see Characteristics of Autonomous Exadata Infrastructure Resources.

Feature	Description
Support for DBMS_CLOUD subprograms	Oracle Autonomous AI Database on Dedicated Exadata Infrastructure now supports many of the subprograms available in the DBMS_CLOUD PL/SQL package. Thus, you can now perform tasks like loading data from files in the cloud, query externally stored data, and export data to other Oracle Database deployments. For more information, see Load Data from Files in the Cloud, Querying External Data with Autonomous AI Database and Exporting Data to Other Oracle Databases. For detailed information about all the supported DBMS_CLOUD subprograms, see DBMS_CLOUD Package Reference.
Store files in directories in your database	Oracle Autonomous AI Database on Dedicated Exadata Infrastructure now provides you the ability to create directories where you can store files in your database, much like you can do in an OS file system. For more information, see Creating and Managing Directories.
Support for work requests	Oracle Autonomous AI Database on Dedicated Exadata Infrastructure is now integrated with the Oracle Cloud Infrastructure Work Requests feature. Work requests allow you to monitor long-running operations like the creation of an Autonomous AI Database. For a list of Oracle Autonomous AI Database on Dedicated Exadata Infrastructure 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.
Restart your database in one step	Previously, to restart your database you had to stop it, wait for it to stop, and then start it. Now, you can perform a restart in a single operation. For more information, see Restart a Dedicated Database.
Support for compartment quotas	Oracle Autonomous AI Database on Dedicated Exadata Infrastructure on dedicated infrastructure now supports the compartment quotas feature of Oracle Cloud Infrastructure. Using this feature you can restrict, on a compartment by compartment basis, the availability of dedicated infrastructure resources in your tenancy. For more information, see Compartment Quotas to Manage Resource Availability. For general information about the compartment quotas feature, see Compartment Quotas.

January 2020

Feature	Description
Custom maintenance schedules for Autonomous Container Databases	Previously, the maintenance schedule for an Autonomous Container Database was determined by the maintenance schedule of the Autonomous Exadata Infrastructure resource hosting it. Now, you can specify a custom maintenance schedule for an Autonomous Container Database. For more information, see Change the Maintenance Schedule of an Autonomous Container Database.

New Feature Announcements (2019)

Announcements for the noteworthy changes made to Autonomous AI Database on Dedicated Exadata Infrastructure in the year 2019.

September 2019

Feature	Description
Autonomous AI Lakehouse supports Dedicated Exadata Infrastructure deployments	When you create an Autonomous AI Lakehouse deployment, you can now choose between the Serverless and dedicated deployment options. For information about creating, managing, and using dedicated Autonomous AI Lakehouse deployments, see <i>User's Guide to Oracle Autonomous AI Lakehouse Dedicated Deployments</i> .
Clone a database	You can now create a new Autonomous AI Database by cloning an existing one, choosing to clone either the full database or only the database metadata. For instructions, see <i>Clone a Dedicated Autonomous AI Database</i> .
Monitor databases using Performance Hub	The Oracle Cloud Infrastructure console now includes Performance Hub for Autonomous AI Databases, which you can use to view real-time and historical performance data. For more information, see <i>Use Performance Hub to Monitor Databases</i> .
Use network security groups to control network access	You can now use network security groups in addition to or instead of security lists to control network access to the VCN and subnets hosting your Autonomous AI Database on Dedicated Exadata Infrastructure deployments. For information about specifying network security groups for your Autonomous Exadata Infrastructure resources, see <i>Change the Network Security Groups Assigned to an Autonomous Exadata Infrastructure Resource</i> . For general information about network security groups, see Network Security Groups , and for a comparison of network security groups and security lists, see Comparison of Security Lists and Network Security Groups .

August 2019

Feature	Description
Use Events and Notifications to watch resource activity	Autonomous AI Database on Dedicated Exadata Infrastructure now generates and sends events to the Event service when it performs operations on resources. 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 listings of the operations that generate events, see Autonomous AI Database Event Types , Autonomous Container Database Event Types , and Autonomous Exadata Infrastructure Event Types .

July 2019

Feature	Description
Use Enterprise Manager to manage databases	You can now use Oracle Enterprise Manager to discover and manage dedicated Autonomous AI Transaction Processing databases, provided you are using, at a minimum, Enterprise Manager version 13.3 with EM DB Plugin Bundle Patch 13.3.2.0.190731. For information on how to take advantage of this capability, see <i>Enterprise Manager Cloud Control Administrator's Guide for Oracle Autonomous AI Databases</i>.

June 2019

A new deployment choice for Autonomous AI Transaction Processing, called Dedicated Deployment, is now available.

Now you have two deployment choices, Serverless and Dedicated.

- **Serverless** — a simple and elastic deployment choice. Oracle autonomously operates all aspects of the database life cycle from database placement to backup and updates.
- **Dedicated** — a private cloud in public cloud deployment choice. A completely dedicated compute, storage, network and database service for only a single tenant. Dedicated deployment provides for the highest levels of security isolation and governance. The customer has customizable operational policies for autonomous operations: workload placement, workload optimization, update scheduling, availability level, over provisioning and peak usage.

Dedicated deployment allows an organization to rethink database IT, enabling a customizable private database cloud in the public cloud. The dedicated choice makes it possible to deliver a self-service database capability that aligns with organizational structure. Different lines of business or project teams can have complete autonomy in their individual execution while the company itself gets a fleet wide simplified aggregation of overall health, availability and cost management. This separation of Fleet versus Database administration allows simple budgeting controls and resource isolation without getting in the way of the line of business execution. And a dedicated database deployment will support the entire spectrum of needs from simple apps to apps that require the highest governance, consistent performance and operational controls.

Autonomous AI Transaction Processing Dedicated Features

- Private cloud isolation: DB, Container DB, Cluster of VM, Hardware (Exadata), Network (VCN)
- Customizable operational policies: DB separation, patching and upgrade, availability, over-provisioning and peak usage
- Configurable: maintenance schedule and type, backup retention time
- Deploy in Oracle Cloud VCN with Private IPs
- Oracle Database 19c: Auto Indexing and other 19c enhancements
- Separate Administrative Roles: Fleet Admin and DB Admin
- Others: Performance Hub Metrics, Database Vault, SQL Developer Web, ORDS, Application Express, Enterprise Manager

Visit the Autonomous AI Transaction Processing documentation [home page](#) for more information.

Autonomous AI Database Supplied Package Reference

DBMS_CLOUD Package

DBMS_CLOUD for Access Management

This section covers the DBMS_CLOUD subprograms used for access management.

Prerequisites

As a developer, you can use DBMS_CLOUD procedures with Autonomous AI Databases deployed on Oracle Public Cloud, Multicloud or Exadata Cloud@Customer.

Depending on the deployment choice, the following prerequisites must be met to use the DBMS_CLOUD procedures with Amazon S3, Azure Blob Storage, and Google Cloud Storage service providers.

-
- [Oracle Public Cloud or Multicloud](#)
 - [Exadata Cloud@Customer](#)

Oracle Public Cloud or Multicloud

An outbound connectivity must have been configured using a NAT gateway, by your fleet administrator as described below:

- Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
- After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 1. Go to the **Subnet Details** page for the subnet.
 2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
 3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** NAT Gateway
 - **Target:** The name of the NAT gateway you just created in the VCNIf such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.
 4. Return to the **Subnet Details** page for the subnet.
 5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.

6. In the side menu, under **Resources**, click **Egress Rules**.
7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** All

If such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

DBMS_CLOUD Subprograms for Access Management

The subprograms for credential management within the DBMS_CLOUD package, including creating, deleting, and updating credentials.

Subprogram	Description
CREATE_CREDENTIAL Procedure	This procedure stores cloud service credentials in Autonomous AI Database.
DROP_CREDENTIAL Procedure	This procedure removes an existing credential from Autonomous AI Database.
UPDATE_CREDENTIAL Procedure	This procedure updates cloud service credential attributes in Autonomous AI Database.

CREATE_CREDENTIAL Procedure

This procedure stores cloud service credentials in Autonomous AI Database.

Use stored cloud service credentials to access the cloud service for data loading, for querying external data residing in the cloud, or for other cases when you use DBMS_CLOUD procedures with a `credential_name` parameter.

Syntax

```
DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name  IN VARCHAR2,
    username         IN VARCHAR2,
    password         IN VARCHAR2 DEFAULT NULL);
```

```
DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name IN VARCHAR2,
    user_ocid      IN VARCHAR2,
    tenancy_ocid   IN VARCHAR2,
    private_key    IN VARCHAR2,
    fingerprint    IN VARCHAR2);
```

Parameters

Parameter	Description
<code>credential_name</code>	The name of the credential to be stored. The <code>credential_name</code> parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens.
<code>username</code>	The username and password arguments together specify your cloud service credentials. See the usage notes for what to specify for the username and password for different cloud services.
<code>password</code>	The username and password arguments together specify your cloud service credentials.
<code>user_ocid</code>	Specifies the user's OCID. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the User's OCID.
<code>tenancy_ocid</code>	Specifies the tenancy's OCID. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the Tenancy's OCID.
<code>private_key</code>	Specifies the generated private key. Private keys generated with a passphrase are not supported. You need to generate the private key without a passphrase. See How to Generate an API Signing Key for details on generating a key pair in PEM format.
<code>fingerprint</code>	Specifies a fingerprint. After a generated public key is uploaded to the user's account the fingerprint is displayed in the console. Use the displayed fingerprint for this argument. See How to Get the Key's Fingerprint and How to Generate an API Signing Key for more details.

Usage Notes

- This operation stores the credentials in the database in an encrypted format.

- You can see the credentials in your schema by querying the `user_credentials` table.
- The ADMIN user can see all the credentials by querying the `dba_credentials` table.
- You only need to create credentials once unless your cloud service credentials change. Once you store the credentials you can then use the same credential name for DBMS_CLOUD procedures that require a `credential_name` parameter.
- This procedure is overloaded. If you provide one of the key based authentication attributes, `user_ocid`, `tenancy_ocid`, `private_key`, or `fingerprint`, the call is assumed to be an Oracle Cloud Infrastructure Signing Key based credential.
- You can list credentials from the view `ALL_CREDENTIALS`. For example, run the following command to list credentials:

```
SELECT credential_name, username, comments FROM all_credentials;
```

Oracle Cloud Infrastructure Credentials (Auth Tokens)

For Oracle Cloud Infrastructure the username is your Oracle Cloud Infrastructure user name. The password is your Oracle Cloud Infrastructure auth token. See [Working with Auth Tokens](#).

For example:

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

Use Auth Token based credentials when you are authenticating calls to OCI Object Storage. For calls to any other type of Oracle Cloud Infrastructure cloud service, use Oracle Cloud Infrastructure Signing Key Based Credentials.

Oracle Cloud Infrastructure Signing Key Based Credentials

Use the Oracle Cloud Infrastructure signing key related parameters, including: `user_ocid`, `tenancy_ocid`, `private_key`, and `fingerprint` with Oracle Cloud Infrastructure Signing Keys authentication.

For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'OCI_KEY_CRED',
    user_ocid      =>
'ocid1.user.oc1..aaaaaaaauq54mi7zdyfhw33ozkwuontjceel7fok5nq3bf2vwetkppqsoa',
    tenancy_ocid   =>
'ocid1.tenancy.oc1..aabbbaafcue47pqmrf4vigneebgbcmmy5r7xvoypicjqgge32ewnc
yx2a',
    private_key    =>
'MIIEogIBAAKCAQEAtUnxbmrekwgVac6FdWeRzoXvIpA9+0r1....wtNpESQQQ0QLGPD8NM//
JEBg=',
    fingerprint    => 'f2:db:f9:18:a4:aa:fc:94:f4:f6:6c:39:96:16:aa:27');
```

```
END;
/
```

Private keys generated with a passphrase are not supported. You need to generate the private key without a passphrase. See [How to Generate an API Signing Key](#) for more information.

Amazon Web Services (AWS) Credentials

If your source files reside in Amazon S3 or you are calling an AWS API, the username is your AWS access key ID and the password is your AWS secret access key. See [AWS Identity and Access Management](#).

Microsoft Azure Credentials

If your source files reside in Azure Blob Storage or you are calling an Azure API, the username is your Azure storage account name and the password is an Azure storage account access key. See [About Azure storage accounts](#).

Amazon S3-Compatible Credentials

Service	Credentials Information
Google Cloud Storage	If your source files reside in Google Cloud Storage or you are calling Google Cloud Storage APIs, then you need to set a default Google project and obtain an HMAC key to create credentials to supply with Google Cloud Storage S3-compatible URLs. Use the HMAC key id as the username, and the HMAC secret as the password. See Projects and HMAC Keys for more information.

Manage DBMS_CLOUD Credential to authenticate to C3 Object Storage

You can use the DBMS_CLOUD.CREATE_CREDENTIAL procedure to provide the Autonomous AI Database with the authentication information required to connect to the C3 Object Storage Bucket. Use the Oracle Cloud Infrastructure signing-key related parameters, including user_ocid, tenancy_ocid, private_key, and fingerprint with Oracle Cloud Infrastructure Signing Keys authentication.

For example,

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'OBJECT_STORAGE_CREDENTIAL',
    user_ocid       => 'ocid1.user.oc1..abcdedfxxxxx.....',
    tenancy_ocid    => 'ocid1.tenancy.oc1..abcdefxyyyyyy....',
    private_key     =>
'MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBAgEAAoIBAQDku_Rest_of_PrivateKey_no_s
paces_no_newlines',
    fingerprint     => '3d:ba:f9:aa:bb:cc:dd:ee:ff:gg:hh:ii:jj:xa:xb:xc'
  );
END;
/
```

DROP_CREDENTIAL Procedure

This procedure removes an existing credential from Autonomous AI Database.

Syntax

```
DBMS_CLOUD.DROP_CREDENTIAL (  
    credential_name    IN VARCHAR2);
```

Parameters

Parameter	Description
credential_name	The name of the credential to be removed.

UPDATE_CREDENTIAL Procedure

This procedure updates an attribute with a new value for a specified credential_name.

Use stored credentials for data loading, for querying external data residing in the Cloud, or wherever you use DBMS_CLOUD procedures with a credential_name parameter.

Syntax

```
DBMS_CLOUD.UPDATE_CREDENTIAL (  
    credential_name    IN VARCHAR2,  
    attribute          IN VARCHAR2,  
    value              IN VARCHAR2);
```

Parameters

Parameter	Description
credential_name	The name of the credential to be updated.
attribute	Name of attribute to update. For a username/password type credential, the valid attribute values are: USERNAME and PASSWORD. See CREATE_CREDENTIAL Procedure for more information.
value	New value for the specified attribute.

Usage Notes

- The username value is case sensitive. It cannot contain double quotes or spaces.
- The ADMIN user can see all the credentials by querying dba_credentials.
- You only need to create credentials once unless your cloud service credentials change. Once you store the credentials you can then use the same credential name for DBMS_CLOUD procedures that require a credential_name parameter.
- You can list credentials from the view ALL_CREDENTIALS. For example, run the following command to list credentials:

```
SELECT credential_name, username, comments FROM all_credentials;
```

Examples

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

```
BEGIN
  DBMS_CLOUD.UPDATE_CREDENTIAL(
    credential_name => 'ARN_CRED',
    attribute => 'aws_role_arn',
    value => 'NEW_AWS_ARN');
END;
/
```

DBMS_CLOUD for Objects and Files

This section covers the DBMS_CLOUD subprograms used to work with objects and files.

Prerequisites

As a developer, you can use DBMS_CLOUD procedures with Autonomous AI Databases deployed on Oracle Public Cloud, Multicloud or Exadata Cloud@Customer.

Depending on the deployment choice, the following prerequisites must be met to use the DBMS_CLOUD procedures with Amazon S3, Azure Blob Storage, and Google Cloud Storage service providers.

-
- [Oracle Public Cloud or Multicloud](#)
 - [Exadata Cloud@Customer](#)

Oracle Public Cloud or Multicloud

An outbound connectivity must have been configured using a NAT gateway, by your fleet administrator as described below:

- Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
- After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 1. Go to the **Subnet Details** page for the subnet.
 2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.

3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** NAT Gateway
 - **Target:** The name of the NAT gateway you just created in the VCNIf such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.
4. Return to the **Subnet Details** page for the subnet.
5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
6. In the side menu, under **Resources**, click **Egress Rules**.
7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** AllIf such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

DBMS_CLOUD Subprograms for Objects and Files

The subprograms for object and file management within the DBMS_CLOUD package.

Subprogram	Description
COPY_COLLECTION Procedure	This procedure loads data into existing SODA collection either from Cloud Object Storage or from files in a directory.
COPY_DATA Procedure	This procedure loads data into existing Autonomous AI Database tables either from Cloud Object Storage or from files in a directory.
COPY_DATA Procedure for Avro, ORC, or Parquet Files	This procedure with the format parameter type set to the value orc, parquet, or avro loads data into existing Autonomous AI Database tables from ORC, Parquet, or Avro files in the Cloud or from ORC, Parquet, or Avro files in a directory. Similar to text files, the data is copied from the source ORC, Parquet, or Avro file into the preexisting internal table.
COPY_OBJECT Procedure	This procedure copies files from one Cloud Object Storage bucket to another.
CREATE_EXTERNAL_TABLE Procedure	This procedure creates an external table on files in the Cloud or on files in a directory. This allows you to run queries on external data from Autonomous AI Database.
CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files	This procedure with the format parameter type set to the value parquet, orc, or avro, creates an external table with either Parquet, ORC, or Avro format files in the Cloud or in a directory. This allows you to run queries on external data from Autonomous AI Database.
CREATE_EXTERNAL_TEXT_INDEX Procedure	This procedure creates text index on the object store files.
CREATE_HYBRID_PART_TABLE Procedure	This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous AI Database.
DELETE_ALL_OPERATIONS Procedure	This procedure clears either all data load operations logged in the user_load_operations table in your schema or clears all the data load operations of the specified type, as indicated with the type parameter.
DELETE_FILE Procedure	This procedure removes the specified file from the specified directory on Autonomous AI Database
DELETE_OBJECT Procedure	This procedure deletes the specified object on object store.
DELETE_OPERATION Procedure	This procedure accepts an operation_id as input and deletes the logs associated with the specified operation_id.
DROP_EXTERNAL_TEXT_INDEX Procedure	This procedure drops text index on the object store files.
EXPORT_DATA Procedure	This procedure exports data from Autonomous AI Database to files in the Cloud based on the result of a query. The overloaded form enables you to use the operation_id parameter. Depending on the format parameter type option specified, the procedure exports rows to the Cloud Object store as text with options of CSV, JSON, Parquet, or XML
GET_OBJECT Procedure and Function	This procedure is overloaded. The procedure form reads an object from Cloud Object Storage and copies it to Autonomous AI Database. The function form reads an object from Cloud Object Storage and returns a BLOB to Autonomous AI Database.

Subprogram	Description
LIST_FILES Function	This function lists the files in the specified directory. The results include the file names and additional metadata about the files such as file size in bytes, creation timestamp, and the last modification timestamp.
LIST_OBJECTS Function	This function lists objects in the specified location on object store. The results include the object names and additional metadata about the objects such as size, checksum, creation timestamp, and the last modification timestamp.
MOVE_OBJECT Procedure	This procedure moves an object from one Cloud Object Storage bucket to another one.
PUT_OBJECT Procedure	This procedure is overloaded. In one form the procedure copies a file from Autonomous AI Database to the Cloud Object Storage. In another form the procedure copies a BLOB from Autonomous AI Database to the Cloud Object Storage.
SYNC_EXTERNAL_PART_TABLE Procedure	This procedure simplifies updating an external partitioned table from files in the Cloud. Run this procedure whenever new partitions are added or when partitions are removed from the Object Store source for the external partitioned table.
VALIDATE_EXTERNAL_TABLE Procedure	This procedure validates the source files for an external table, generates log information, and stores the rows that do not match the format options specified for the external table in a <i>badfile</i> table on Autonomous AI Database.
VALIDATE_EXTERNAL_PART_TABLE Procedure	This procedure validates the source files for an external partitioned table, generates log information, and stores the rows that do not match the format options specified for the external table in a <i>badfile</i> table on Autonomous AI Database.
VALIDATE_HYBRID_PART_TABLE Procedure	This procedure validates the source files for a hybrid partitioned table, generates log information, and stores the rows that do not match the format options specified for the hybrid table in a <i>badfile</i> table on Autonomous AI Database.

COPY_COLLECTION Procedure

This procedure loads data into a SODA collection from Cloud Object Storage or from a directory. If the specified SODA collection does not exist, the procedure creates it. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.COPY_COLLECTION (
    collection_name    IN VARCHAR2,
    credential_name    IN VARCHAR2 DEFAULT NULL,
    file_uri_list      IN CLOB,
    format             IN CLOB      DEFAULT NULL
);

DBMS_CLOUD.COPY_COLLECTION (
    collection_name    IN VARCHAR2,
    credential_name    IN VARCHAR2 DEFAULT NULL,
    file_uri_list      IN CLOB,
    format             IN CLOB      DEFAULT NULL,
```

```
        operation_id      OUT NOCOPY NUMBER
    );
```

Parameters

Parameter	Description
collection_name	The name of the SODA collection into which data will be loaded. If a collection with this name already exists, the specified data will be loaded, otherwise a new collection is created.
credential_name	The name of the credential to access the Cloud Object Storage. This parameter is not used when you specify a directory with file_uri_list.

Parameter	Description
file_uri_list	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to FALSE. When the <code>regexuri</code> parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. For example: <pre>file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/[a-z]+[1-3])?.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition. Directory You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: <code>'MY_DIR:filename.ext'</code>. By default the directory name <code>MY_DIR</code> is a database object and is case-insensitive. The file name is case sensitive. You can use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character. For example: <code>'MY_DIR: *'</code> or <code>'MY_DIR:test?'</code> To specify multiple directories, use a comma separated list of directories: For example: <code>'MY_DIR1: *, MY_DIR2:test?'</code> Use double quotes to specify a case-sensitive directory name. For example: <code>'"my_dir1": *, "my_dir2":Test?'</code> To include a quote character, use two quotes. For example: <code>'MY_DIR: ' 'filename.ext'</code>. This specifies the filename starts with a quote (').
format	The options describing the format of the source files. These options are specified as a JSON string. Supported formats are: <code>characterset</code>, <code>compression</code>, <code>ignoreblanklines</code>, <code>jsonpath</code>, <code>maxdocsize</code>, <code>recorddelimiter</code>, <code>rejectlimit</code>, <code>unpackarray</code>, <code>keyassignment</code>, and <code>keypath</code>. Apart from the mentioned formats for JSON data, Autonomous AI Database supports other formats too. For the list of format arguments supported by Autonomous AI Database, see Format Parameter.

Parameter	Description
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'OBJ_STORE_CRED',
        username        => 'user_name@oracle.com',
        password        => 'password'
    );

    DBMS_CLOUD.COPY_COLLECTION(
        collection_name => 'myCollection',
        credential_name => 'OBJ_STORE_CRED',
        file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/adbexample/b/json/o/myCollection.json'
    );
END;
/

```

COPY_DATA Procedure

This procedure loads data into existing Autonomous AI Database tables from files in the Cloud, or from files in a directory. The overloaded form enables you to use the operation_id parameter.

Syntax

```

DBMS_CLOUD.COPY_DATA (
    table_name      IN VARCHAR2,
    credential_name IN VARCHAR2 DEFAULT NULL,
    file_uri_list   IN CLOB,
    schema_name     IN VARCHAR2,
    field_list      IN CLOB,
    format          IN CLOB);

```

```

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

```

Parameters

Parameter	Description
table_name	The name of the target table on the database. The target table needs to be created before you run COPY_DATA.
credential_name	The name of the credential to access the Cloud Object Storage. This parameter is not used when you specify a directory with file_uri_list.
file_uri_list	You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Cloud source file URIs This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. Regular expressions can only be used when the regexuri format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. For example: <pre>file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/[a-z]+[1-3])?.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on REGEXP_LIKE condition. Directory You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: 'MY_DIR: filename.ext'. By default the directory name MY_DIR is a database object and is case-insensitive. The file name is case sensitive. You can use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character. For example: 'MY_DIR: *' or 'MY_DIR: test?' To specify multiple directories, use a comma separated list of directories: For example: 'MY_DIR1: *, MY_DIR2: test?' Use double quotes to specify a case-sensitive directory name. For example: '"my_dir1": *, "my_dir2": Test?' To include a quote character, use two quotes. For example: 'MY_DIR: ' ' filename.ext'. This specifies the filename starts with a quote (').

Parameter	Description
schema_name	The name of the schema where the target table resides. The default value is NULL meaning the target table is in the same schema as the user running the procedure.
field_list	Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the <i>column_list</i> parameter. This argument's syntax is the same as the <i>field_list</i> clause in regular Oracle external tables. For more information see <i>field_list</i> in Oracle Database 19c Utilities or Oracle Database 26ai Utilities . When the format parameter type option value is json, this parameter is ignored. For an example using field_list, see CREATE EXTERNAL TABLE Procedure .
format	The options describing the format of the source, log, and bad files. For the list of the options and how to specify the values see Format Parameter. For Avro, ORC, or Parquet file format options, see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Usage Note

The default record delimiter is detected newline. With detected newline, DBMS_CLOUD tries to automatically find the correct newline character to use as the record delimiter. DBMS_CLOUD first searches for the Windows newline character \r\n. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, DBMS_CLOUD searches for the UNIX/Linux newline character \n, and if it finds one it uses \n as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "KUP-04020: found record longer than buffer size supported". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.

See Format Parameter for information on the recorddelimiter format option.

Examples

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

```
BEGIN
    DBMS_CLOUD.COPY_DATA(
        table_name => 'CHANNELS',
        credential_name => 'DEF_CRED_NAME',
        file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/channels.txt',
    );
END;
```

```

        format => json_object('delimiter' value ',')
    );
END;
/

BEGIN
    DBMS_CLOUD.COPY_DATA(
        table_name      => 'ORDERS',
        schema_name     => 'TEST_SCHEMA',
        credential_name => 'DEF_CRED_NAME',
        file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/adbexample/b/json/o/orde[r]s.tbl.1'
        format          => json_object('ignoreblanklines' value TRUE,
                                      'rejectlimit' value '0',
                                      'dateformat' value 'yyyy-mm-dd',
                                      'regexuri' value TRUE)
    );
END;
/

```

COPY_DATA Procedure for Avro, ORC, or Parquet Files

This procedure with the format parameter type set to the value avro, orc, or parquet loads data into existing Autonomous AI Database tables from Avro, ORC, or Parquet files in the Cloud or from files in a directory.

Similar to text files, the data is copied from the source Avro, ORC, or Parquet file into the preexisting internal table.

Syntax

```

DBMS_CLOUD.COPY_DATA (
    table_name      IN VARCHAR2,
    credential_name IN VARCHAR2 DEFAULT NULL,
    file_uri_list   IN CLOB,
    schema_name     IN VARCHAR2 DEFAULT,
    field_list      IN CLOB DEFAULT,
    format          IN CLOB DEFAULT);

```

Parameters

Parameter	Description
table_name	The name of the target table on the database. The target table needs to be created before you run COPY_DATA.
credential_name	The name of the credential to access the Cloud Object Storage. This parameter is not used when you specify a directory with file_uri_list.

Parameter	Description
file_uri_list	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to FALSE. When the <code>regexuri</code> parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. For example: <pre>file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/[a-z]+[1-3])?.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition. Directory You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: <code>'MY_DIR: filename.ext'</code>. By default the directory name <code>MY_DIR</code> is a database object and is case-insensitive. The file name is case sensitive. You can use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character. For example: <code>'MY_DIR: *'</code> or <code>'MY_DIR: test?'</code> To specify multiple directories, use a comma separated list of directories: For example: <code>'MY_DIR1: *, MY_DIR2: test?'</code> Use double quotes to specify a case-sensitive directory name. For example: <code>'"my_dir1": *, "my_dir2": Test?'</code> To include a quote character, use two quotes. For example: <code>'MY_DIR: ' 'filename.ext'</code>. This specifies the filename starts with a quote (').
schema_name	The name of the schema where the target table resides. The default value is NULL meaning the target table is in the same schema as the user running the procedure.

Parameter	Description
field_list	Ignored for Avro, ORC, or Parquet files. The fields in the source match the external table columns by name. Source data types are converted to the external table column data type. For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping. For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details on mapping. For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details on mapping.
format	The options describing the format of the source files. For Avro, ORC, or Parquet files, only two options are supported: see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.

Usage Notes

- As with other data files, Avro, ORC, and Parquet data loads generate logs that are viewable in the tables `dba_load_operations` and `user_load_operations`. Each load operation adds a record to `dba[user]_load_operations` that indicates the table containing the logs.
The log table provides summary information about the load.
- For Avro, ORC, or Parquet, when the `format` parameter type is set to the value `avro`, `orc`, or `parquet`, the `BADFILE_TABLE` table is always empty.
 - For Parquet files, `PRIMARY KEY` constraint errors throw an ORA error.
 - If data for a column encounters a conversion error, for example, the target column is not large enough to hold the converted value, the value for the column is set to `NULL`. This does not produce a rejected record.

COPY_OBJECT Procedure

This procedure copies an object from one Cloud Object Storage bucket or folder to another.

The source and target bucket or folder can be in the same or different Cloud Object store provider.

When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, you can give separate credential names for the source and target locations.

The source credential name is by default also used by the target location when target credential name is not provided.

Syntax

```
DBMS_CLOUD.COPY_OBJECT (
    source_credential_name IN VARCHAR2 DEFAULT NULL,
    source_object_uri      IN VARCHAR2,
    target_object_uri      IN VARCHAR2,
    target_credential_name IN VARCHAR2 DEFAULT NULL
);
```

Parameters

Parameter	Description
source_credential_name	The name of the credential to access the source Cloud Object Storage. If you do not supply a source_credential_name value, the credential_name is set to NULL.
source_object_uri	Specifies URI, that point to the source Object Storage bucket or folder location. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See Cloud Object Storage URI Formats for more information.
target_object_uri	Specifies the URI for the target Object Store. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See Cloud Object Storage URI Formats for more information.
target_credential_name	The name of the credential to access the target Cloud Object Storage location. If you do not supply a target_credential_name value, the target_object_uri is set to the source_credential_name value.

Example

```

BEGIN
DBMS_CLOUD.COPY_OBJECT (
    source_credential_name => 'OCI_CRED',
    source_object_uri      => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/bgfile.csv',
    target_object_uri      => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/myfile.csv'
);
END;
/

```

CREATE_EXTERNAL_PART_TABLE Procedure

This procedure creates an external partitioned table on files in the Cloud, or from files in a directory. This allows you to run queries on external data from Autonomous AI Database.

Syntax

```

DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (
    table_name          IN VARCHAR2,
    credential_name      IN VARCHAR2,
    partitioning_clause IN CLOB,
    column_list          IN CLOB,
    field_list           IN CLOB DEFAULT,
    format               IN CLOB DEFAULT);

```

```

DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE (
    table_name          IN VARCHAR2,
    credential_name      IN VARCHAR2,

```



```

file_uri_list    IN VARCHAR2,
column_list     IN CLOB,
field_list      IN CLOB DEFAULT,
format          IN CLOB DEFAULT);

```

Parameters

Parameter	Description
table_name	The name of the external table.
credential_name	The name of the credential to access the Cloud Object Storage.
partitioning_clause	Specifies the complete partitioning clause, including the location information for individual partitions. If you use the <code>partitioning_clause</code> parameter, the <code>file_uri_list</code> parameter is not allowed.
file_uri_list	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to <code>TRUE</code>. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to <code>FALSE</code>. When the <code>regexuri</code> parameter is set to <code>TRUE</code> the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. This option is only supported with external tables that are created on a file in the Object Storage. For example: <pre>file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]???.csv'</pre> If you use the parameter <code>file_uri_list</code>, the <code>partitioning_clause</code> parameter is not allowed. The format of the URIs depends on the Cloud Object Storage service. See Cloud Object Storage URI Formats for more information. See REGEXP_LIKE Condition in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition.

Parameter	Description
<code>column_list</code>	Comma-delimited list of column names and data types for the external table. This parameter has the following requirements, depending on the type of the data files specified with the <code>file_uri_list</code> parameter: • The <code>column_list</code> parameter is required with unstructured files. Using unstructured files, for example with CSV text files, the <code>column_list</code> parameter must specify all the column names and data types inside the data file as well as the partition columns derived from the object name. • The <code>column_list</code> parameter is optional with structured files. For example, with Avro, ORC, or Parquet data files, the <code>column_list</code> is not required. When the <code>column_list</code> is not included, the format parameter <code>partition_columns</code> option must include specifications for both column names (name) and data types (type).
<code>field_list</code>	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 <code>column_list</code> parameter. This argument's syntax is the same as the <code>field_list</code> clause in regular Oracle external tables. For more information see <i>field_list</i> in Oracle Database 19c Utilities or Oracle Database 26ai Utilities.

Parameter	Description
format	The format option <code>partition_columns</code> specifies the <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> column names and data types of partition columns when the partition columns are derived from the file path, depending on the type of data file, structured or unstructured: - When the <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> includes the <code>column_list</code> parameter and the data files are unstructured, such as with CSV text files, <code>partition_columns</code> does not include the data type. For example, use a format such as the following for this type of <code>partition_columns</code> specification: <pre>"partition_columns":["state","zipcode"]'</pre> The data type is not required because it is specified in the <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> <code>column_list</code> parameter. - When the <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> does not include the <code>column_list</code> parameter and the data files are structured, such as Avro, ORC, or Parquet files, the <code>partition_columns</code> option includes both the column name, name sub-clause, and the data type, type sub-clause. For example, the following shows a <code>partition_columns</code> specification: <pre>"partition_columns":[{"name":"country", "type":"varchar2(10)"}, {"name":"year", "type":"number"}, {"name":"month", "type":"varchar2(10)"}]'</pre> If the data files are unstructured and the type sub-clause is specified with <code>partition_columns</code>, the type sub-clause is ignored. For object names that are not based on hive format, the order of the <code>partition_columns</code> specified columns must match the order as they appear in the object name in the file path specified in the <code>file_uri_list</code> parameter. To see all the format parameter options describing the format of the source files, see Cloud Object Storage URI Formats.

Usage Notes

- You cannot call this procedure with both `partitioning_clause` and `file_uri_list` parameters.
- Specifying the `column_list` parameter is optional with structured data files, including Avro, Parquet, or ORC data files. If `column_list` is not specified, the format parameter `partition_columns` option must include both name and type.
- The `column_list` parameter is required with unstructured data files, such as CSV text files.
- The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in the supported cloud object storage services, including:
 - Oracle Cloud Infrastructure Object Storage
 - Azure Blob Storage

See Cloud Object Storage URI Formats for more information.

- The procedure `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` supports external partitioned files in directories, either in a local file system or in a network file system.
- When you call `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` with the `file_uri_list` parameter, the types for columns specified in the Cloud Object Store file name must be one of the following types:

```

VARCHAR2(n)
NUMBER(n)
NUMBER(p, s)
NUMBER
DATE
TIMESTAMP(9)

```

- The default record delimiter is detected newline. With detected newline, `DBMS_CLOUD` tries to automatically find the correct newline character to use as the record delimiter. `DBMS_CLOUD` first searches for the Windows newline character `\r\n`. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, `DBMS_CLOUD` searches for the UNIX/Linux newline character `\n`, and if it finds one it uses `\n` as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "KUP-04020: found record longer than buffer size supported". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.

See Cloud Object Storage URI Formats for information on the `recorddelimiter` format option.

- The external partitioned tables you create with `DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from.
 - `file$path`: Specifies the file path text up to the beginning of the object name.
 - `file$name`: Specifies the object name, including all the text that follows the bucket name.

Examples

Example using the `partitioning_clause` parameter:

```

BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name => 'PET1',
    credential_name => 'OBJ_STORE_CRED',
    format => json_object('delimiter' value ',', 'recorddelimiter' value
'newline', 'character set' value 'us7ascii'),
    column_list => 'col1 number, col2 number, col3 number',
    partitioning_clause => 'partition by range (col1)
                          (partition p1 values less than (1000) location
                           ( '&base_URL//file_11.txt' )
                           ,
                           partition p2 values less than (2000) location
                           ( '&base_URL//file_21.txt' )
                           ,
                           partition p3 values less than (3000)
                           location

```

```

                                ( '&base_URL/file_31.txt' )
                                )'
                                );
END;
/

BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name      => 'PET',
    format          => json_object('delimiter' value ' '),
    column_list     => 'name varchar2(20), gender varchar2(10), salary
number',
    partitioning_clause => 'partition by range (salary)
      ( -- Use test1.csv in the DEFAULT DIRECTORY DATA_PUMP_DIR
        partition p1 values less than (100) LOCATION
(''test1.csv''),
          -- Use test2.csv in a specified directory MY_DIR
        partition p2 values less than (300) DEFAULT DIRECTORY
MY_DIR LOCATION (''test2.csv'')          )' );
END;
/

```

Example using the `file_uri_list` and `column_list` parameters with unstructured data files:

```

BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name => 'MYSALES',
    credential_name => 'DEF_CRED_NAME',
    file_uri_list  => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/*.csv',
    column_list   => 'product varchar2(100), units number, country varchar2(100), year
number, month varchar2(2)',
    field_list    => 'product, units', --[Because country, year and month are not in the
file, they are not listed in the field list]
    format        => '{"type":"csv", "partition_columns":["country","year","month"]}');
END;
/

```

Example using the `file_uri_list` without the `column_list` parameter with structured data files:

```

BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name => 'MYSALES',
    credential_name => 'DEF_CRED_NAME',
    DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
      table_name      => 'MYSALES',
      credential_name => 'DEF_CRED_NAME',
      file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-
string/b/bucketname/o/*.parquet',
      format          =>
        json_object('type' value 'parquet', 'schema' value 'first',
          'partition_columns' value
            json_array(

```

```

        json_object('name' value 'country', 'type' value
'varchar2(100)'),
        json_object('name' value 'year', 'type' value 'number'),
        json_object('name' value 'month', 'type' value 'varchar2(2)')
    )
);
END;
/

```

CREATE_EXTERNAL_PART_TABLE Procedure for C3 Object Storage

You can use the DBMS_CLOUD package to create external partitioned tables using the objects stored in the C3 Object Storage Bucket.

For example:

```

BEGIN
  DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE(
    table_name => 'USAGE_EXTP',
    credential_name => 'OBJECT_STORAGE_CREDENTIAL',
    file_uri_list => 'https://objectstorage.mydomain.com/n/mytenancy/b/
mybucket/o/usage-data/*usage-data*.parquet',
    format => json_object(
      'type' value 'parquet', 'schema' value 'first',
      'partition_columns' value json_array(
        json_object('name' value 'year', 'type' value
'number'),
        json_object('name' value 'month', 'type' value 'number'),
        json_object('name' value 'day', 'type' value 'number')
      )
    )
  );
END;
/

```

CREATE_EXTERNAL_TABLE Procedure

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

Syntax

```

DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
  table_name      IN VARCHAR2,
  credential_name IN VARCHAR2,
  file_uri_list   IN CLOB,
  column_list     IN CLOB,
  field_list      IN CLOB DEFAULT,
  format          IN CLOB DEFAULT);

```

Parameters

Parameter	Description
table_name	The name of the external table.
credential_name	The name of the credential to access the Cloud Object Storage. This parameter is not used when you specify a directory or a Table Hyperlink URL with <code>file_uri_list</code> .
file_uri_list	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to FALSE. When the <code>regexuri</code> parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. This option is only supported with external tables that are created on a file in the Object Storage. For example: <pre>file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]???.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition. Directory You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: <code>'MY_DIR:filename.ext'</code>. By default the directory name <code>MY_DIR</code> is a database object and is case-insensitive. The file name is case sensitive. You can use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character. For example: <code>'MY_DIR: *'</code> or <code>'MY_DIR: test?'</code> To specify multiple directories, use a comma separated list of directories: For example: <code>'MY_DIR1: *, MY_DIR2: test?'</code> Use double quotes to specify a case-sensitive directory name. For example: <code>'"my_dir1": *, "my_dir2": Test?'</code> To include a quote character, use two quotes. For example: <code>'MY_DIR: ' 'filename.ext'</code>. This specifies the filename starts with a quote (').

Parameter	Description
column_list	Comma-delimited list of column names and data types for the external table.
field_list	Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the column_list parameter. This argument's syntax is the same as the field_list clause in regular Oracle Database external tables. For more information about field_list see <i>ORACLE_LOADER Access Driver field_list under field_definitions Clause</i> in Oracle Database 19c Utilities or Oracle Database 26ai Utilities .
format	The options describing the format of the source files. For the list of the options and how to specify the values see Format Parameter. For Avro, ORC, or Parquet format files, see CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files .

Usage Notes

- The procedure DBMS_CLOUD.CREATE_EXTERNAL_TABLE supports external partitioned files in the supported cloud object storage services including Azure Blob Storage.

The credential is a table level property; therefore, the external files must be on the same object store.

See Cloud Object Storage URI Formats for more information.
- The default record delimiter is detected newline. With detected newline, DBMS_CLOUD tries to automatically find the correct newline character to use as the record delimiter. DBMS_CLOUD first searches for the Windows newline character \r\n. If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, DBMS_CLOUD searches for the UNIX/Linux newline character \n, and if it finds one it uses \n as the record delimiter for all files in the procedure. If the source files use a combination of different record delimiters, you may encounter an error such as, "KUP-04020: found record longer than buffer size supported". In this case, you need to either modify the source files to use the same record delimiter or only specify the source files that use the same record delimiter.

See Format Parameter for information on the recorddelimiter format option.

Example

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



```

        column_list => 'REPORT_DATE DATE,
                        REPORT_DATE_COPY DATE,
                        ACTUAL_MEAN_TEMP NUMBER,
                        ACTUAL_MIN_TEMP NUMBER,
                        ACTUAL_MAX_TEMP NUMBER,
                        AVERAGE_MIN_TEMP NUMBER,
                        AVERAGE_MAX_TEMP NUMBER,
                        AVERAGE_PRECIPITATION NUMBER');
    END;
/

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

```

CREATE_EXTERNAL_TABLE Procedure for C3 Object Storage

You can use the DBMS_CLOUD package to create external tables using the objects stored in the C3 Object Storage Bucket.

For example:

```

BEGIN
    DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
        table_name => 'CHANNELS_EXT',
        credential_name => 'OBJECT_STORAGE_CREDENTIAL',
        file_uri_list => '<https://objectstorage.mydomain.com/n/mytenancy/b/
mybucket/o/channels.txt>',
        format => json_object('delimiter' value ','),
        column_list => 'CHANNEL_ID NUMBER, CHANNEL_DESC VARCHAR2(20),
CHANNEL_CLASS VARCHAR2(20)'
    );
END;
/

```

CREATE_EXTERNAL_TABLE Procedure for Apache Iceberg

This procedure creates external tables for Apache Iceberg tables in the supported configurations.

These specific configurations are supported:

- Iceberg tables on AWS:
 - Iceberg tables registered with AWS Glue Data Catalog, created with Spark or Athena. For more information see [Use the AWS Glue connector to read and write Apache Iceberg tables with ACID transactions and perform time travel](#) and [Using Iceberg tables](#).
 - Iceberg tables stored on AWS S3 by directly providing the URL for the root metadata file.
- Iceberg tables on OCI:
 - Iceberg tables generated with OCI Data Flow using a Hadoop Catalog. For more information, see [Oracle Data Flow Examples](#) and [Using a Hadoop Catalog](#).
 - Iceberg tables stored on OCI Object Storage by directly providing the URL for the root metadata file.

Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name      IN VARCHAR2,
    credential_name IN VARCHAR2 DEFAULT NULL,
    file_uri_list   IN CLOB,
    column_list     IN CLOB DEFAULT NULL,
    field_list      IN CLOB DEFAULT NULL,
    format          IN CLOB DEFAULT NULL
);
```

Parameters

Parameter	Description
table_name	The name of the external table.
credential_name	The name of the credential used to access the data files, the metadata files and the Iceberg Catalog (if used). For AWS and OCI configurations, the credential should be created as described in CREATE_CREDENTIAL Procedure. AWS Amazon Resource Names (ARN) credentials are currently not supported.
file_uri_list	Must be NULL if an Iceberg catalog is specified (see format parameter below). If an iceberg catalog is not used, then the file_uri_list must contain the URI to the iceberg metadata file.
column_list	Must be NULL, as the column names and types are automatically derived from Iceberg metadata. The column names match the names found in the underlying data files (Parquet, Avro, ORC). The Oracle data types are derived using the Parquet/Avro/ORC mappings between Iceberg and the Parquet, Avro and ORC data types. Therefore users cannot specify the column_list.
field_list	Must be NULL, as column names and data types are automatically derived from the Iceberg metadata.
format	The format parameter has a different structure depending on the type of Iceberg table, and what information is used to create the external table, for example information from a data catalog or a direct metadata URI. See Iceberg Support on OCI Data Flow Samples , and DBMS_CLOUD URI Formats for more information.

Example format parameter for AWS Iceberg tables using an AWS Glue Catalog

A sample format parameter when creating tables over an AWS Iceberg table using an AWS Glue Catalog is as follows:

```
format => json_object('access_protocol' value
    json_object('protocol_type' value 'iceberg',
        'protocol_config' value
            json_object('iceberg_catalog_type' value 'aws_glue',
                'iceberg_glue_region' value 'glue region',
                'iceberg_table_path' value
                    'database_name.table_name')));
```

Where, the `access_protocol` parameter contains a JSON object with two elements as follows:

- `protocol_type`: Must be 'iceberg'
- `protocol_config`: A nested JSON object specifying the iceberg catalog details.
 - `iceberg_catalog_type`: Must be 'aws_glue'
 - `iceberg_glue_region`: The catalog region, e.g. 'us-west-1'
 - `iceberg_table_path`: A *glue database.glue table name* path.

Example format parameter for AWS Iceberg table using a metadata file URI

A sample format parameter when creating tables over an AWS Iceberg table using a metadata file URI, is as follows:

```
format => json_object('access_protocol' value
                     json_object('protocol_type' value 'iceberg'))
```

Example format parameter for OCI Iceberg table using HadoopCatalog catalog

A sample format parameter when creating tables over an OCI Iceberg table created by OCI Data Flow using HadoopCatalog catalog, is as follows:

```
format => json_object('access_protocol' value
                     json_object('protocol_type' value 'iceberg',
                                'protocol_config' value
                                json_object('iceberg_catalog_type' value 'hadoop',
                                           'iceberg_warehouse' value '<OCI folder
URI>',
                                           'iceberg_table_path' value
'<database_name.table_name>')));
```

Where, the `access_protocol` parameter contains a JSON object with two elements as follows:

- `protocol_type`: Must be 'iceberg'
- `protocol_config`: A nested JSON object specifying the iceberg catalog details.
 - `iceberg_catalog_type`: Must be 'hadoop'
 - `iceberg_warehouse`: The warehouse directory path used when generating the table, in native URI format.
 - `iceberg_table_path`: The *database_name.table name* path used when creating the table.

Example format parameter for OCI Iceberg table using the URI of the metadata file

A sample format parameter when creating tables over an OCI Iceberg table using the URI of the metadata file, is as follows:

```
format => json_object('access_protocol' value
                     json_object('protocol_type' value 'iceberg'))
```

Where the `access_protocol` parameter contains a JSON object with one element as follows:

- `protocol_type`: Must be 'iceberg'

CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files

This procedure with the `format` parameter type set to the value `avro`, `orc`, or `parquet` creates an external table with either Avro, ORC, or Parquet format files in the Cloud or in a directory.

This allows you to run queries on external data from Autonomous AI Database.

Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (  
    table_name      IN VARCHAR2,  
    credential_name IN VARCHAR2 DEFAULT NULL,  
    file_uri_list   IN CLOB,  
    column_list     IN CLOB,  
    field_list      IN CLOB DEFAULT,  
    format          IN CLOB DEFAULT);
```

Parameters

Parameter	Description
<code>table_name</code>	The name of the external table.
<code>credential_name</code>	The name of the credential to access the Cloud Object Storage. This parameter is not used when you specify a directory with <code>file_uri_list</code> .

Parameter	Description
<code>file_uri_list</code>	This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to FALSE. When the <code>regexuri</code> parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. This option is only supported with external tables that are created on a file in the Object Storage. For example: <pre>file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]???.parquet'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on REGEXP_LIKE condition. Directory You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: <code>'MY_DIR:filename.ext'</code>. By default the directory name <code>MY_DIR</code> is a database object and is case-insensitive. The file name is case sensitive. You can use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, the character "?" can be used as the wildcard for a single character. For example: <code>'MY_DIR: *'</code> or <code>'MY_DIR:test?'</code> To specify multiple directories, use a comma separated list of directories: For example: <code>'MY_DIR1:*, MY_DIR2:test?'</code> Use double quotes to specify a case-sensitive directory name. For example: <code>'"my_dir1":*, "my_dir2":Test?'</code> To include a quote character, use two quotes. For example: <code>'MY_DIR: ' 'filename.ext'</code>. This specifies the filename starts with a quote (').

Parameter	Description
column_list	(Optional) This field, when specified, overrides the format -> schema parameter which specifies that the schema, columns, and data types, are derived automatically. See the format parameter for details. When the column_list is specified for Avro, ORC, or Parquet source, the column names must match those columns found in the file. Oracle data types must map appropriately to the Avro, ORC, or Parquet data types. For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details. For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping for details. For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details.
field_list	Ignored for Avro, ORC, or Parquet files. The fields in the source match the external table columns by name. Source data types are converted to the external table column data type. For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details. For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details.
format	For Avro, ORC, or Parquet type source files, see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet for details.

Examples ORC

```
format => '{"type":"orc", "schema": "all"}'
```

```
format => json_object('type' value 'orc', 'schema' value 'first')
```

Examples Avro

```
format => '{"type":"avro", "schema": "all"}'
```

```
format => json_object('type' value 'avro', 'schema' value 'first')
```

Examples Parquet

```
format => '{"type":"parquet", "schema": "all"}'
```

```
format => json_object('type' value 'parquet', 'schema' value 'first')
```

Avro, ORC, or Parquet Column Name Mapping to Oracle Column Names

See DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping for information on column name mapping and column name conversion usage in Oracle SQL.

CREATE_EXTERNAL_TEXT_INDEX Procedure

This procedure creates a text index on Object Storage files.

The CREATE_EXTERNAL_TEXT_INDEX procedure creates text index on the Object Storage files specified at the `location_uri` location. The index is refreshed at regular intervals, for any new additions or deletions done with files on location URI.

Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX (  
    credential_name IN VARCHAR2 DEFAULT NULL,  
    location_uri    IN VARCHAR2,  
    index_name      IN VARCHAR2,  
    format          IN CLOB          DEFAULT NULL  
);
```

Parameters

Parameter	Description
<code>credential_name</code>	The name of the credential to access the Cloud Object Storage location. For public, pre-authenticated, or pre-signed bucket URIs, a NULL can be specified. If you do not supply a <code>credential_name</code> value, the <code>credential_name</code> is set to a NULL value.

Parameter	Description
location_uri	This parameter specifies Comma-delimited list of source file URIs. You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. This parameter is mandatory. Cloud source file URIs This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to <code>TRUE</code>. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to <code>FALSE</code>. When the <code>regexuri</code> parameter is set to <code>TRUE</code> the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. Example using regular expressions: <pre>location_uri => 'https:// objectstorage.my\$region.oraclecloud.com/n/namespace- string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+ (/)*[a-z]+[1-3]???.csv'</pre> Example using wild cards: <pre>location_uri => 'https:// objectstorage.my\$region.oraclecloud.com/n/namespace- string/b/bucketname/o/year=????/month=??/*.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <code>REGEXP_LIKE</code> Condition for more information on <code>REGEXP_LIKE</code> condition.
index_name	Specifies the name of the index you are building on the files located at the <code>location_uri</code> location. This parameter is mandatory.

Parameter	Description
format	Specifies additional configuration options. Options are specified as a JSON string. The supported format options are: refresh_rate: Specifies the frequency in minutes at which the local index is refreshed. New file uploads and deletions result in an index refresh. The default value is 5 minutes. binary_files: Specifies if the contents of the files to be indexed are binary. For example, PDF, MS-Word, The default value is FALSE. stop_words: Specifies a list of stop words can be supplied when you create indexes. The stop_words value indicates if it is a list of stop words or a table of stop words. When a JSON array is provided the stop words parameter it is treated as a list, otherwise the stop words parameter is treated as a table name whose column "STOP_WORDS" is used to read in the list of stop words. You can specify stop words using the following methods: - JSON Array: For example: format := '{"stop_words": ["king", "queen"]}' - Stop word table name: For example: format := '{"stop_words": "STOP_WORDS_TABLE"}' If you do not supply a format parameter, the format is set to a NULL value.

Example

```

BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TEXT_INDEX (
    credential_name => 'DEFAULT_CREDENTIAL',
    location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespaces-string/b/bucketname/o/ts_data/'
    index_name      => 'EMP',
    format          => JSON_OBJECT ('refresh_rate' value 10)
);
END;
/

```

CREATE_HYBRID_PART_TABLE Procedure

This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous AI Database using database objects and files in the Cloud, or database objects and files in a directory.

Syntax

```

DBMS_CLOUD.CREATE_HYBRID_PART_TABLE (
    table_name          IN VARCHAR2,
    credential_name     IN VARCHAR2,
    partitioning_clause IN CLOB,
    column_list         IN CLOB,
    field_list          IN CLOB DEFAULT,
    format              IN CLOB DEFAULT);

```

Parameters

Parameter	Description
<code>table_name</code>	The name of the external table.
<code>credential_name</code>	The name of the credential to access the Cloud Object Storage.
<code>partitioning_clause</code>	Specifies the complete partitioning clause, including the location information for individual partitions. To use directories, the partitioning clause supports the <code>LOCATION</code> and <code>DEFAULT DIRECTORY</code> values. You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to <code>TRUE</code>. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to <code>FALSE</code>. When the <code>regexuri</code> parameter is set to <code>TRUE</code> the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. Regular expression patterns are not supported for directory names. For example: <pre>partitioning_clause => 'partition by range (col1) (partition p1 values less than (1000) external location ('https://objectstorage.us- phoenix-1.oraclecloud.com/n/namespace-string/b/ bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z] +[1-3]??.txt'), ...</pre> See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition.
<code>column_list</code>	Comma-delimited list of column names and data types for the external table.
<code>field_list</code>	Identifies the fields in the source files and their data types. The default value is <code>NULL</code> meaning the fields and their data types are determined by the <code>column_list</code> parameter. This argument's syntax is the same as the <code>field_list</code> clause in regular Oracle external tables. For more information see <i>field_list</i> in Oracle Database 19c Utilities or Oracle Database 26ai Utilities .
<code>format</code>	The options describing the format of the source files. For the list of the options and how to specify the values see Format Parameter.

Usage Notes

- The procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` supports external partitioned files in the supported cloud object storage services including Azure Blob Storage.

The credential is a table level property; therefore, the external files must be on the same object store.

See Cloud Object Storage URI Formats for more information.

- The procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` supports hybrid partitioned files in directories, either in a local file system or in a network file system.
- The external partitioned tables you create with `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` include two invisible columns `file$path` and `file$name`. These columns help identify which file a record is coming from.
 - `file$path`: Specifies the file path text up to the beginning of the object name.
 - `file$name`: Specifies the object name, including all the text that follows the bucket name.

Examples

```

BEGIN
  DBMS_CLOUD.CREATE_HYBRID_PART_TABLE(
    table_name => 'HPT1',
    credential_name => 'OBJ_STORE_CRED',
    format => json_object('delimiter' value ',', 'recorddelimiter' value
'newline', 'character set' value 'us7ascii'),
    column_list => 'col1 number, col2 number, col3 number',
    partitioning_clause => 'partition by range (col1)
                          (partition p1 values less than (1000)
external location
                          ( '&base_URL/file_11.txt' )
                          ,
                          partition p2 values less than (2000)
external location
                          ( '&base_URL/file_21.txt' )
                          ,
                          partition p3 values less than (3000)
                          )'
    );
END;
/

BEGIN
  DBMS_CLOUD.CREATE_HYBRID_PART_TABLE(
    table_name => 'HPT1',
    format => json_object('delimiter' value ',', 'recorddelimiter' value
'newline'),
    column_list => 'NAME VARCHAR2(30), GENDER VARCHAR2(10), BALANCE number',
    partitioning_clause => 'partition by range (B 2 ALANCE)
                          (partition p1 values less than (1000) external DEFAULT
DIRECTORY DATA_PUMP_DIR LOCATION (''Scott_male_1000.csv''),
                          partition p2 values less than (2000) external DEFAULT
DIRECTORY DATA_PUMP_DIR LOCATION (''Mary_female_3000.csv''),
                          partition p3 values less than (3000))' );
END;
/

```

DELETE_ALL_OPERATIONS Procedure

This procedure clears either all data load operations logged in the `user_load_operations` table in your schema or clears all the data load operations of the specified type, as indicated with the `type` parameter.

Syntax

```
DBMS_CLOUD.DELETE_ALL_OPERATIONS (  
    type          IN VARCHAR DEFAULT NULL);
```

Parameters

Parameter	Description
type	Specifies the type of operation to delete. Type values can be found in the TYPE column in the user_load_operations table. If no type is specified all rows are deleted.

Usage Note

- DBMS_CLOUD.DELETE_ALL_OPERATIONS does not delete currently running operations (operations in a "Running" status).

DELETE_FILE Procedure

This procedure removes the specified file from the specified directory on Autonomous AI Database.

Syntax

```
DBMS_CLOUD.DELETE_FILE (  
    directory_name  IN VARCHAR2,  
    file_name       IN VARCHAR2,  
    force           IN BOOLEAN DEFAULT FALSE);
```

Parameters

Parameter	Description
directory_name	The name of the directory on the Autonomous AI Database instance.
file_name	The name of the file to be removed.
force	Ignore and do not report errors if the file does not exist. Valid values are: TRUE and FALSE. The default value is FALSE.

Note

To run DBMS_CLOUD.DELETE_FILE with a user other than ADMIN you need to grant write privileges on the directory that contains the file to that user. For example, run the following command as ADMIN to grant write privileges to atpc_user:

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

Example

```

BEGIN
  DBMS_CLOUD.DELETE_FILE(
    directory_name => 'DATA_PUMP_DIR',
    file_name => 'exp1.dmp' );
END;
/

```

DELETE_OBJECT Procedure

This procedure deletes the specified object on object store.

Syntax

```

DBMS_CLOUD.DELETE_OBJECT (
  credential_name      IN VARCHAR2,
  object_uri           IN VARCHAR2,
  force                IN BOOLEAN DEFAULT FALSE);

```

Parameters

Parameter	Description
credential_name	The name of the credential to access the Cloud Object Storage.
object_uri	Object or file URI for the object to delete. The format of the URI depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.
force	Ignore and do not report errors if object does not exist. Valid values are: TRUE and FALSE. The default value is FALSE.

Example

```

BEGIN
  DBMS_CLOUD.DELETE_OBJECT(
    credential_name => 'DEF_CRED_NAME',
    object_uri => 'https://objectstorage.us-ashburn-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/exp1.dmp' );
END;
/

```

DELETE_OPERATION Procedure

This procedure clears the data load entries for the specified operation ID logged in the user_load_operations or dba_load_operations tables in your schema.

Syntax

```

DBMS_CLOUD.DELETE_OPERATION (
  id      IN NUMBER);

```

Parameters

Parameter	Description
id	Specifies the operation ID associated with the log file entries you want to delete.

Example

```
SELECT id FROM user_load_operations WHERE type LIKE '%BAD%';  
EXEC DBMS_CLOUD.DELETE_OPERATION(id);
```

DROP_EXTERNAL_TEXT_INDEX Procedure

This procedure drops text index on the Object Storage files.

The `DROP_EXTERNAL_TEXT_INDEX` procedure drops the specified index created with the `CREATE_EXTERNAL_TEXT_INDEX` procedure.

Syntax

```
DBMS_CLOUD.DROP_EXTERNAL_TEXT_INDEX (  
    index_name      IN  VARCHAR2,  
);
```

Parameters

Parameter	Description
index_name	Specifies the name of the index you are dropping. The index name must match the name provided at the time of the index creation. This parameter is mandatory.

Example

```
BEGIN  
DBMS_CLOUD.DROP_EXTERNAL_TEXT_INDEX (  
    index_name => 'EMP',  
);  
END;  
/
```

EXPORT_DATA Procedure

This procedure exports data from Autonomous AI Database based on the result of a query. This procedure is overloaded and supports writing files to the cloud or to a directory.

Based on the `format` type parameter, the procedure exports files to the Cloud or to a directory location as text files in CSV, JSON, Parquet, or XML format.

Syntax

```
DBMS_CLOUD.EXPORT_DATA (
    credential_name IN VARCHAR2 DEFAULT NULL,
    file_uri_list   IN CLOB,
    format          IN CLOB,
    query           IN CLOB);
```

```
DBMS_CLOUD.EXPORT_DATA (
    credential_name IN VARCHAR2 DEFAULT NULL,
    file_uri_list   IN CLOB DEFAULT NULL,
    format          IN CLOB DEFAULT NULL,
    query           IN CLOB DEFAULT NULL,
    operation_id    OUT NOCOPY NUMBER);
```

Parameters

Parameter	Description
<code>credential_name</code>	The name of the credential to access the Cloud Object Storage. When the credential parameter is not included, this specifies output to a directory.
<code>file_uri_list</code>	There are different forms, depending on the value of the format parameter and depending on whether you include a credential parameter: - When the format parameter type value is <code>json</code>: The JSON on Object Store or to the specified directory location is saved with a generated file name based on the value of the <code>file_uri_list</code> parameter. See File Naming for Text Output (CSV,JSON, Parquet, or XML) for more information. - When the format parameter type value is <code>datapump</code>, the <code>file_uri_list</code> is a comma-delimited list of the dump files. This specifies the files to be created on the Object Store. Use of wildcard and substitution characters is not supported in the <code>file_uri_list</code>. - When the <code>credential_name</code> parameter is not specified you provide a directory name in <code>file_uri_list</code>. The format of the URIs depend on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.
<code>format</code>	A JSON string that provides export format options. Supported option is: <code>type</code>: The type format option is required and must have one of the values: <code>csv</code> <code>datapump</code> <code>json</code> <code>parquet</code> <code>xml</code>. See DBMS_CLOUD Package Format Options for EXPORT_DATA.

Parameter	Description
query	Use this parameter to specify a SELECT statement so that only the required data is exported. The query determines the contents of the files you export as text files CSV, JSON, Parquet, or XML, or as dump files. For example: <pre>SELECT warehouse_id, quantity FROM inventories</pre> When the format type value is json, each query result is checked and if it is not JSON, as determined with the function: <code>JSON_OBJECT_T.parse()</code>, <code>DBMS_CLOUD.EXPORT_DATA</code> transforms the query to include <code>JSON_OBJECT</code> function to convert the row into JSON. See <i>JSON_OBJECT</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference and <i>JSON_OBJECT_T Object Type</i> in Oracle Database 19c PL/SQL Packages and Types Reference or Oracle Database 26ai PL/SQL Packages and Types Reference for more information. For example: <pre>SELECT JSON_OBJECT(* RETURNING CLOB) from(SELECT warehouse_id, quantity FROM inventories)</pre>
operation_id	Use this parameter to track the progress and final status of the export operation as the corresponding ID in the <code>USER_LOAD_OPERATIONS</code> view.

Usage Notes:

- The query parameter value that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Depending on the format parameter specified, `DBMS_CLOUD.EXPORT_DATA` outputs the results of the specified query on the Cloud Object Store or to a directory location in CSV, JSON, Parquet, or XML file formats.

See *Export Data to Object Store as Text Using `DBMS_CLOUD.EXPORT_DATA`* and *Export data to a Directory Using `DBMS_CLOUD.EXPORT_DATA`* for more information on using `DBMS_CLOUD.EXPORT_DATA` with CSV, JSON, Parquet, or XML output files.

- For CSV, JSON, or XML output, by default when a generated file contains 10MB of data a new output file is created. However, if you have less than 10MB of result data you may have multiple output files, depending on the database service and the number of ECPUs (OCPUs if your database uses OCPUs) for the Autonomous AI Database instance.

See *File Naming for Text Output (CSV, JSON, Parquet, or XML)* for more information.

The default output file chunk size is 10MB for CSV, JSON, or XML. You can change this value with the format parameter `maxfilesize` option. See *DBMS_CLOUD Package Format Options for `EXPORT_DATA`* for more information.

- For CSV, JSON, or XML output, by default when a generated file contains 10MB of data a new output file is created. However, if you have less than 10MB of result data you may have multiple output files, depending on the database service and the number of ECPUs (OCPUs if your database uses OCPUs) for the Autonomous AI Database instance.

See *File Naming for Text Output (CSV, JSON, Parquet, or XML)* for more information.

- The parquet output format can only be used with high and medium database services for Autonomous AI Databases on Dedicated Exadata Infrastructure.

Usage Notes for `DBMS_CLOUD.EXPORT_DATA` with Output to a Directory

- The provided directory must exist and you must be logged in as the ADMIN user or have WRITE access to the directory.

- DBMS_CLOUD.EXPORT_DATA does not create directories.
- The procedure does not overwrite files. For example, if a dump file in the `file_uri_list` exists, DBMS_CLOUD.EXPORT_DATA reports an error such as:

```
ORA-31641: unable to create dump file "/u02/exports/123.dmp"
ORA-27038: created file already exists
```

Examples

The following example shows DBMS_CLOUD.EXPORT_DATA with the `format` type parameter with the value `json`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'OBJ_STORE_CRED',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp1.json',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'json', 'compression' value
'gzip'));
  );
END;
/
```

The following example shows DBMS_CLOUD.EXPORT_DATA with the `format` type parameter with the value `xml`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'OBJ_STORE_CRED',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp1.xml',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'xml', 'compression' value
'gzip'));
  );
END;
/
```

The following example shows DBMS_CLOUD.EXPORT_DATA with the `format` type parameter with the value `csv`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'OBJ_STORE_CRED',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp.csv',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'csv', 'delimiter' value
'|', 'compression' value 'gzip', 'header' value true ));
  );
END;
/
```

GET_OBJECT Procedure and Function

This procedure is overloaded. The procedure form reads an object from Cloud Object Storage and copies it to Autonomous AI Database. The function form reads an object from Cloud Object Storage and returns a BLOB to Autonomous AI Database.

Syntax

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

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

Parameters

Parameter	Description
credential_name	The name of the credential to access the Cloud Object Storage.
object_uri	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.
directory_name	The name of the directory on the database. 1
file_name	Specifies the name of the file to create. If file name is not specified, the file name is taken from after the last slash in the object_uri parameter. For special cases, for example when the file name contains slashes, use the file_name parameter.
startoffset	The offset, in bytes, from where the procedure starts reading.
endoffset	The offset, in bytes, until where the procedure stops reading.
compression	Specifies the compression used to store the object. When compression is set to 'AUTO' the file is uncompressed (the value 'AUTO' implies the object specified with object_uri is compressed with Gzip).

1

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;
```

Return Values

The function form reads from Object Store and `DBMS_CLOUD.GET_OBJECT` returns a `BLOB`.

Examples

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

To read character data from a file in Object Store:

```
SELECT to_clob(
  DBMS_CLOUD.GET_OBJECT(
    credential_name => 'OBJ_STORE_CRED',
    object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/file.txt'))
FROM DUAL;
```

To add an image stored on Object Store in a `BLOB` in the database:

```
DECLARE
  l_blob BLOB := NULL;
BEGIN
  l_blob := DBMS_CLOUD.GET_OBJECT(
    credential_name => 'OBJ_STORE_CRED',
    object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/MyImage.gif' );
END;
/
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

LIST_FILES Function

This function lists the files in the specified directory. The results include the file names and additional metadata about the files such as file size in bytes, creation timestamp, and the last modification timestamp.

Syntax

```
DBMS_CLOUD.LIST_FILES (  
    directory_name      IN VARCHAR2)  
    RETURN TABLE;
```

Parameters

Parameter	Description
directory_name	The name of the directory on the database. This parameter supports wild cards.

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`:


```
GRANT READ ON DIRECTORY data_pump_dir TO atpc_user;
```
- This is a pipelined table function with return type as `DBMS_CLOUD_TYPES.list_object_ret_t`.
- `DBMS_CLOUD.LIST_FILES` does not obtain the checksum value and returns `NULL` for this field.
- `DBMS_CLOUD.LIST_FILES` supports filtering files selectively using the `directory:filename` syntax. Wildcards are supported in the filename.

Example:

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

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

OBJECT_NAME	BYTES	CHECKSUM	CREATED	LAST_MODIFIED
-----	-----	-----	-----	-----

cwallet.sso	2965		2018-12-12T18:10:47Z	
2019-11-23T06:36:54Z				

Example queries using wild cards:

You can use `DBMS_CLOUD.LIST_FILES` with `DIRECTORY:FILE` syntax, either by specifying a file or using wild cards. `*` and `?` are the supported wild card characters.

Example 1: List all CSV files starting with "f" in the directory DATA_PUMP_DIR using a wild card.

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('DATA_PUMP_DIR:f*.csv');
```

Example 2: List file f1.csv in the directory DATA_PUMP_DIR specifying the file name.

```
SELECT * FROM DBMS_CLOUD.LIST_FILES('DATA_PUMP_DIR:f1.csv');
```

LIST_OBJECTS Function

This function lists objects in the specified location on object store. The results include the object names and additional metadata about the objects such as size, checksum, creation timestamp, and the last modification timestamp.

Syntax

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

Parameters

Parameter	Description
credential_name	The name of the credential to access the Cloud Object Storage.
location_uri	Object storage bucket or folder URI. The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. Wild cards are supported to selectively list files or subfolders in a URI. The characters "*" and "?" are considered wild card characters. Example using wild cards: location_uri => 'https://objectstorage.my\$region.oraclecloud.com/n/namespace-string/b/bucketname/o/year=????/month=??/*.csv'

Usage Notes

- Depending on the capabilities of the object store, DBMS_CLOUD.LIST_OBJECTS does not return values for certain attributes and the return value for the field is NULL in this case.
- All supported Object Stores return values for the OBJECT_NAME, BYTES, and CHECKSUM fields.
- The following table shows support for the fields CREATED and LAST_MODIFIED by Object Store:

Object Store	CREATED	LAST_MODIFIED
Oracle Cloud Infrastructure Native	Returns timestamp	Returns timestamp
Oracle Cloud Infrastructure Swift	Returns NULL	Returns timestamp
Amazon S3	Returns NULL	Returns timestamp

Object Store	CREATED	LAST_MODIFIED
Amazon S3-Compatible	Returns NULL	Returns timestamp
Azure	Returns timestamp	Returns timestamp

- The checksum value is the MD5 checksum. This is a 32-character hexadecimal number that is computed on the object contents.
- This is a pipelined table function with return type as `DBMS_CLOUD_TYPES.list_object_ret_t`.

Example

This is a pipelined function that returns a row for each object. For example, use the following query to use this function:

```
SELECT * FROM DBMS_CLOUD.LIST_OBJECTS('OBJ_STORE_CRED',
    'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
    bucketname/o/');
```

```
OBJECT_NAME  BYTES          CHECKSUM
CREATED      LAST_MODIFIED
-----
-----
cwallet.sso  2965           2339a2731ba24a837b26d344d643dc07
2019-11-23T06:36:54Z
```

In this example, *namespace-string* is the Oracle Cloud Infrastructure object storage namespace and *bucketname* is the bucket name. See [Understanding Object Storage Namespaces](#) for more information.

MOVE_OBJECT Procedure

This procedure moves an object from one Cloud Object Storage bucket or folder to another.

The source and target bucket or folder can be in the same or different Cloud Object store provider.

When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, you can give separate credential names for the source and target locations.

The source credential name is by default also used by the target location when target credential name is not provided.

Syntax

```
DBMS_CLOUD.MOVE_OBJECT (
    source_credential_name IN VARCHAR2 DEFAULT NULL,
    source_object_uri      IN VARCHAR2,
    target_object_uri      IN VARCHAR2,
    target_credential_name IN VARCHAR2 DEFAULT NULL
);
```

Parameters

Parameter	Description
source_credential_name	The name of the credential to access the source Cloud Object Storage. If you do not supply a source_credential_name value, the credential_name is set to NULL.
source_object_uri	Specifies URI, that point to the source Object Storage bucket or folder location. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See Cloud Object Storage URI Formats for more information.
target_object_uri	Specifies the URI for the target Object Storage bucket or folder, where the files need to be moved. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See Cloud Object Storage URI Formats for more information.
target_credential_name	The name of the credential to access the target Cloud Object Storage location. If you do not supply a target_credential_name value, the target_object_uri is set to the source_credential_name value.

Example

```

BEGIN
DBMS_CLOUD.MOVE_OBJECT (
    source_credential_name => 'OCI_CRED',
    source_object_uri      => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/bgfile.csv',
    target_object_uri      => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/myfile.csv'
);
END;
/

```

PUT_OBJECT Procedure

This procedure is overloaded. In one form the procedure copies a file from Autonomous AI Database to the Cloud Object Storage. In another form the procedure copies a BLOB from Autonomous AI Database to the Cloud Object Storage.

Syntax

```

DBMS_CLOUD.PUT_OBJECT (
    credential_name    IN VARCHAR2,
    object_uri         IN VARCHAR2,
    directory_name     IN VARCHAR2,
    file_name          IN VARCHAR2,
    compression        IN VARCHAR2 DEFAULT NULL);

```

```

DBMS_CLOUD.PUT_OBJECT (
    credential_name    IN VARCHAR2,

```

```

object_uri      IN VARCHAR2,
contents        IN BLOB
compression     IN VARCHAR2 DEFAULT NULL);

```

Parameters

Parameter	Description
credential_name	The name of the credential to access the Cloud Object Storage.
object_uri	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.
directory_name	The name of the directory on the Autonomous AI Database. 1
contents	Specifies the BLOB to copy from Autonomous AI Database to the Cloud Object Storage.
file_name	The name of the file in the specified directory.
compression	Specifies the compression used to store the object. Default value: NULL

1

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;
```

Example

To handle BLOB data after in-database processing and then store the data directly into a file in the object store:

```

DECLARE
    my_blob_data BLOB;
BEGIN
    /* Some processing producing BLOB data and populating my_blob_data */
    DBMS_CLOUD.PUT_OBJECT(
        credential_name => 'OBJ_STORE_CRED',
        object_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/my_new_file',
        contents => my_blob_data);
END;
/

```

Usage Notes

Depending on your Cloud Object Storage, the size of the object you transfer is limited as follows:

Cloud Object Storage Service	Object Transfer Size Limit
Oracle Cloud Infrastructure Object Storage	50 GB
Amazon S3	5 GB
Azure Blob Storage	256 MB
Amazon S3-Compatible	Set by the object store provider. For more information, refer to the provider's documentation.

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

See Cloud Object Storage URI Formats for more information.

SYNC_EXTERNAL_PART_TABLE Procedure

This procedure simplifies updating an external partitioned table from files in the Cloud. Run this procedure whenever new partitions are added or when partitions are removed from the Object Store source for the external partitioned table.

Syntax

```
DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE (
    table_name      IN VARCHAR2,
    schema_name     IN VARCHAR2 DEFAULT,
    update_columns  IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
<code>table_name</code>	The name of the target table. The target table needs to be created before you run <code>DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE</code> .
<code>schema_name</code>	The name of the schema where the target table resides. The default value is <code>NULL</code> meaning the target table is in the same schema as the user running the procedure.
<code>update_columns</code>	The new files may introduce a change to the schema. Updates supported include: new columns, deleted columns. Updates to existing columns, for example a change in the data type throw errors. Default Value: <code>False</code>

VALIDATE_EXTERNAL_PART_TABLE Procedure

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

Syntax

```
DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
    table_name           IN VARCHAR2,
    partition_name       IN CLOB DEFAULT,
    subpartition_name    IN CLOB DEFAULT,
    schema_name          IN VARCHAR2 DEFAULT,
    rowcount             IN NUMBER DEFAULT,
    partition_key_validation IN BOOLEAN DEFAULT,
    stop_on_error        IN BOOLEAN DEFAULT);
```

```
DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
    table_name           IN VARCHAR2,
    operation_id         OUT NUMBER,
    partition_name       IN CLOB DEFAULT,
    subpartition_name    IN CLOB DEFAULT,
    schema_name          IN VARCHAR2 DEFAULT,
    rowcount             IN NUMBER DEFAULT,
    partition_key_validation IN BOOLEAN DEFAULT,
    stop_on_error        IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
table_name	The name of the external table.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.
partition_name	If defined, then only a specific partition is validated. If not specified then read all partitions sequentially until rowcount is reached.
subpartition_name	If defined, then only a specific subpartition is validated. If not specified then read from all external partitions or subpartitions sequentially until rowcount is reached.
schema_name	The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure.
rowcount	Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.
partition_key_validation	For internal use only. Do not use this parameter.
stop_on_error	Determines if the validate should stop when a row is rejected. The default value is TRUE meaning the validate stops at the first rejected row. Setting the value to FALSE specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the rowcount parameter.

VALIDATE_EXTERNAL_TABLE Procedure

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

Syntax

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

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

Parameters

Parameter	Description
table_name	The name of the external table.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.
schema_name	The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure.
rowcount	Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.
stop_on_error	Determines if the validate should stop when a row is rejected. The default value is TRUE meaning the validate stops at the first rejected row. Setting the value to FALSE specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the rowcount parameter. If the external table refers to Avro, ORC, or Parquet files then the validate stops at the first rejected row. When the external table specifies the format parameter type set to the value avro, orc, or parquet, the parameter stop_on_error effectively always has the value TRUE. Thus, the table badfile will always be empty for an external table referring to Avro, ORC, or Parquet files.

Usage Notes

- DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE works with both partitioned external tables and hybrid partitioned tables. This potentially reads data from all external partitions until rowcount is reached or stop_on_error applies. You do not have control over which partition, or parts of a partition, is read in which order.

VALIDATE_HYBRID_PART_TABLE Procedure

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

Syntax

```
DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
    table_name          IN VARCHAR2,
    partition_name      IN CLOB DEFAULT,
    subpartition_name   IN CLOB DEFAULT,
    schema_name        IN VARCHAR2 DEFAULT,
    rowcount            IN NUMBER DEFAULT,
    partition_key_validation IN BOOLEAN DEFAULT,
    stop_on_error       IN BOOLEAN DEFAULT);
```

```
DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
    table_name          IN VARCHAR2,
    operation_id        OUT NUMBER,
    partition_name      IN CLOB DEFAULT,
    subpartition_name   IN CLOB DEFAULT,
    schema_name        IN VARCHAR2 DEFAULT,
    rowcount            IN NUMBER DEFAULT,
    partition_key_validation IN BOOLEAN DEFAULT,
    stop_on_error       IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
table_name	The name of the external table.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.
partition_name	If defined, then only a specific partition is validated. If not specified then read from all external partitions sequentially until rowcount is reached.
subpartition_name	If defined, then only a specific subpartition is validated. If not specified then read from all external partitions or subpartitions sequentially until rowcount is reached.
schema_name	The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure.
rowcount	Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned.
partition_key_validation	For internal use only. Do not use this parameter.
stop_on_error	Determines if the validate should stop when a row is rejected. The default value is TRUE meaning the validate stops at the first rejected row. Setting the value to FALSE specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the rowcount parameter.

DBMS_CLOUD for Bulk File Management

The subprograms for bulk file operations within the DBMS_CLOUD package.

Subprogram	Description
BULK_COPY Procedure	This procedure copies files from one Cloud Object Storage bucket to another.
BULK_DELETE Procedure	The procedure deletes files from Cloud Object Storage bucket or folder.
BULK_DOWNLOAD Procedure	This procedure downloads files from Cloud Object store bucket to a directory in Autonomous AI Database.
BULK_MOVE Procedure	This procedure moves files from one Cloud Object Storage bucket to another.
BULK_UPLOAD Procedure	This procedure uploads files from a directory in Autonomous AI Database to the Cloud Object Storage.

Prerequisites

As a developer, you can use DBMS_CLOUD procedures with Autonomous AI Databases deployed on Oracle Public Cloud, Multicloud or Exadata Cloud@Customer.

Depending on the deployment choice, the following prerequisites must be met to use the DBMS_CLOUD procedures with Amazon S3, Azure Blob Storage, and Google Cloud Storage service providers.

-
- [Oracle Public Cloud or Multicloud](#)
 - [Exadata Cloud@Customer](#)

Oracle Public Cloud or Multicloud

An outbound connectivity must have been configured using a NAT gateway, by your fleet administrator as described below:

- Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
- After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 1. Go to the **Subnet Details** page for the subnet.
 2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
 3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** NAT Gateway
 - **Target:** The name of the NAT gateway you just created in the VCN

If such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.

4. Return to the **Subnet Details** page for the subnet.
5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
6. In the side menu, under **Resources**, click **Egress Rules**.
7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** All

If such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

BULK_COPY Procedure

This procedure bulk copies files from one Cloud Object Storage bucket to another. The overloaded form enables you to use the `operation_id` parameter.

You can filter the list of files to be deleted using a regular expression pattern compatible with REGEXP_LIKE operator.

The source and target bucket or folder can be in the same or different Cloud Object store provider.

When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, you can give separate credential names for the source and target locations.

The source credential name is by default also used by the target location.

Syntax

```
DBMS_CLOUD.BULK_COPY (  
    source_credential_name IN VARCHAR2 DEFAULT NULL,  
    source_location_uri    IN VARCHAR2,  
    target_location_uri    IN VARCHAR2,  
    target_credential_name IN VARCHAR2 DEFAULT NULL,  
    regex_filter           IN VARCHAR2 DEFAULT NULL,  
    format                 IN CLOB      DEFAULT NULL  
);
```

```
DBMS_CLOUD.BULK_COPY (  
    source_credential_name IN VARCHAR2 DEFAULT NULL,  
    source_location_uri    IN VARCHAR2,  
    target_location_uri    IN VARCHAR2,  
    target_credential_name IN VARCHAR2 DEFAULT NULL,  
    regex_filter           IN VARCHAR2 DEFAULT NULL,  
    format                 IN CLOB      DEFAULT NULL,  
    operation_id           OUT NUMBER  
);
```

Parameters

Parameter	Description
source_credential_name	The name of the credential to access the Cloud Object Storage. If you do not supply a source_credential_name value, the credential_name is set to NULL.

Parameter	Description
source_location_uri	Specifies the URI that points to the source Object Storage bucket or folder location. This parameter is mandatory. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to <code>TRUE</code>. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to <code>FALSE</code>. When the <code>regexuri</code> parameter is set to <code>TRUE</code> the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. Example using regular expressions: <pre>source_location_uri => 'https:// objectstorage.my\$region.oraclecloud.com/n/namespace- string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+ (/)*[a-z]+[1-3]???.csv'</pre> Example using wild cards: <pre>source_location_uri => 'https:// objectstorage.my\$region.oraclecloud.com/n/namespace- string/b/bucketname/o/year=????/month=??/*.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition.
target_location_uri	Specifies the URI for the target Object Storage bucket or folder, where the files need to be copied. This parameter is mandatory. Specifies URI, that point to the source Object Storage bucket or folder location. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See <i>DBMS_CLOUD Package File Cloud Object Storage URI Formats</i> for more information.
target_credential_name	The name of the credential to access the target Cloud Object Storage location. If you do not supply a <code>target_credential_name</code> value, the <code>target_location_uri</code> is set to the <code>source_credential_name</code> value.

Parameter	Description
regex_filter	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with the REGEXP_LIKE operator. If you do not supply a regex_filter value, the regex_filter is set to NULL. See REGEXP_LIKE Condition for more information.
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: • logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation. The default value is 2 days. • logprefix: It accepts a string value that determines the bulk operation status table name prefix string. The operation type is the default value. For BULK_COPY, the default logprefix value is COPYOBJ. • priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and should run faster. It accepts the following values: – HIGH: Determines the number of parallel files handled using the database's ECPU count (OCPU count if your database uses OCPUs) – MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. – LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Usage Notes

- An error is returned when the source and target URI point to the same Object Storage bucket or folder.

Example

```

BEGIN
DBMS_CLOUD.BULK_COPY (
    source_credential_name => 'OCI_CRED',
    source_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/o',
    target_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
    format                 => JSON_OBJECT ('logretention' value 7, 'logprefix' value
'BULKOP')
);

```

```
END;  
/
```

BULK_DELETE Procedure

This procedure bulk deletes files from the Cloud Object Storage. The overloaded form enables you to use the operation_id parameter. You can filter the list of files to be deleted using a regular expression pattern compatible with REGEXP_LIKE operator.

Syntax

```
DBMS_CLOUD.BULK_DELETE(  
    credential_name IN VARCHAR2 DEFAULT NULL,  
    location_uri    IN VARCHAR2,  
    regex_filter    IN VARCHAR2 DEFAULT NULL,  
    format          IN CLOB      DEFAULT NULL  
);  
  
DBMS_CLOUD.BULK_DELETE (  
    credential_name IN VARCHAR2 DEFAULT NULL,  
    location_uri    IN VARCHAR2,  
    regex_filter    IN VARCHAR2 DEFAULT NULL,  
    format          IN CLOB      DEFAULT NULL,  
    operation_id    OUT NUMBER  
);
```

Parameters

Parameter	Description
credential_name	The name of the credential to access the Cloud Object Storage. If you do not supply a credential_name value, the credential_name is set to NULL.

Parameter	Description
location_uri	Specifies URI, that points to an Object Storage location in the Autonomous AI Database. This parameter is mandatory. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to FALSE. When the <code>regexuri</code> parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. For example: <pre>location_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]???.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition. The format of the URIs depends on the Cloud Object Storage service. See <code>DBMS_CLOUD</code> Package File Cloud Object Storage URI Formats for more information.
regex_filter	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with the <code>REGEXP_LIKE</code> operator. If you do not supply a <code>regex_filter</code> value, the <code>regex_filter</code> is set to NULL. See REGEXP_LIKE Condition for more information.

Parameter	Description
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: • logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation . The default value is 2 days. • logprefix: It accepts a string value that determines the bulk operation status table name prefix string. The operation type is the default value. For BULK_DELETE, the default logprefix value is DELETE. • priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and is completed sooner. It accepts the following values: – HIGH: Determines the number of parallel files handled using the database's ECPU count (OCPU count if your database uses OCPUs). – MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. – LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```

BEGIN
DBMS_CLOUD.BULK_DELETE (
    credential_name => 'OCI_CRED',
    location_uri    => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o',
    format          => JSON_OBJECT ('logretention' value 5, 'logprefix'
value 'BULKDEL')
);
END;
/

```

BULK_DOWNLOAD Procedure

This procedure downloads files into an Autonomous AI Database directory from Cloud Object Storage. The overloaded form enables you to use the `operation_id` parameter. You can filter the list of files to be downloaded using a regular expression pattern compatible with REGEXP_LIKE operator.

Syntax

```
DBMS_CLOUD.BULK_DOWNLOAD (
    credential_name IN VARCHAR2 DEFAULT NULL,
    location_uri    IN VARCHAR2,
    directory_name  IN VARCHAR2,
    regex_filter    IN VARCHAR2 DEFAULT NULL,
    format          IN CLOB      DEFAULT NULL
);
```

```
DBMS_CLOUD.BULK_DOWNLOAD (
    credential_name IN VARCHAR2 DEFAULT NULL,
    location_uri    IN VARCHAR2,
    directory_name  IN VARCHAR2,
    regex_filter    IN VARCHAR2 DEFAULT NULL,
    format          IN CLOB      DEFAULT NULL,
    operation_id    OUT NUMBER
);
```

Parameters

Parameter	Description
<code>credential_name</code>	The name of the credential to access the Cloud Object Storage. If you do not supply a <code>credential_name</code> value, the <code>credential_name</code> is set to NULL.
<code>location_uri</code>	Specifies URI, that points to an Object Storage location in the Autonomous AI Database. This parameter is mandatory. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to TRUE. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to FALSE. When the <code>regexuri</code> parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. For example: <pre>location_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+(/)*[a-z]+[1-3]???.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition.

Parameter	Description
directory_name	The name of the directory on the Autonomous AI Database from where you want to download the files. This parameter is mandatory.
regex_filter	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with the REGEXP_LIKE operator. If you do not supply a regex_filter value, the regex_filter is set to NULL. See REGEXP_LIKE Condition for more information.
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: - logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation. The default value is 2 days. - logprefix: It accepts a string value that determines the bulk operation status table name prefix string. For BULK_DOWNLOAD, the default logprefix value is DOWNLOAD. The operation type is the default value. - priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and is completed sooner. It accepts the following values: - HIGH: Determines the number of parallel files handled using the database's ECPU count (OCPU count if your database uses OCPUs). - MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. - LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```

BEGIN
DBMS_CLOUD.BULK_DOWNLOAD (
    credential_name => 'OCI_CRED',
    location_uri    => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o',
    directory_name  => 'BULK_TEST',
    format          => JSON_OBJECT ('logretention' value 7, 'logprefix'
value 'BULKOP')
);
END;
/

```

BULK_MOVE Procedure

This procedure bulk moves files from one Cloud Object Storage bucket or folder to another. The overloaded form enables you to use the `operation_id` parameter.

You can filter the list of files to be deleted using a regular expression pattern compatible with `REGEXP_LIKE` operator.

The source and target bucket or folder can be in the same or different Cloud Object store provider.

When the source and target are in distinct Object Stores or have different accounts with the same cloud provider, you can give separate credential names for the source and target locations.

The source credential name is by default also used by the target location when target credential name is not provided.

The first step in moving files is copying them to the target location, then deleting the source files, once they are successfully copied.

The object is renamed rather than moved if Object Store allows renaming operations between source and target locations.

Syntax

```
DBMS_CLOUD.BULK_MOVE (
    source_credential_name IN VARCHAR2 DEFAULT NULL,
    source_location_uri    IN VARCHAR2,
    target_location_uri    IN VARCHAR2,
    target_credential_name IN VARCHAR2 DEFAULT NULL,
    regex_filter           IN VARCHAR2 DEFAULT NULL,
    format                 IN CLOB      DEFAULT NULL
);
```

```
DBMS_CLOUD.BULK_MOVE (
    source_credential_name IN VARCHAR2 DEFAULT NULL,
    source_location_uri    IN VARCHAR2,
    target_location_uri    IN VARCHAR2,
    target_credential_name IN VARCHAR2 DEFAULT NULL,
    regex_filter           IN VARCHAR2 DEFAULT NULL,
    format                 IN CLOB      DEFAULT NULL,
    operation_id           OUT NUMBER
);
```

Parameters

Parameter	Description
<code>source_credential_name</code>	The name of the credential to access the source Cloud Object Storage. If you do not supply a <code>source_credential_name</code> value, the <code>credential_name</code> is set to <code>NULL</code> .

Parameter	Description
source_location_uri	Specifies the URI that points to the source Object Storage bucket or folder location. This parameter is mandatory. Cloud source file URIs You can use wildcards as well as regular expressions in the file names in Cloud source file URIs. Regular expressions can only be used when the <code>regexuri</code> format parameter is set to <code>TRUE</code>. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to <code>FALSE</code>. When the <code>regexuri</code> parameter is set to <code>TRUE</code> the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the <code>REGEXP_LIKE</code> function. Example using regular expressions: <pre>source_location_uri => 'https:// objectstorage.my\$region.oraclecloud.com/n/namespace- string/b/bucketname/o(/)*year=[0-9]+(/)*month=[0-9]+ (/)*[a-z]+[1-3]???.csv'</pre> Example using wild cards: <pre>source_location_uri => 'https:// objectstorage.my\$region.oraclecloud.com/n/namespace- string/b/bucketname/o/year=????/month=??/*.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on <code>REGEXP_LIKE</code> condition.
target_location_uri	Specifies the URI for the target Object Storage bucket or folder, where the files need to be moved. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.
target_credential_name	The name of the credential to access the target Cloud Object Storage location. If you do not supply a <code>target_credential_name</code> value, the <code>target_location_uri</code> is set to the <code>source_credential_name</code> value.
regex_filter	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with the <code>REGEXP_LIKE</code> operator. If you do not supply a <code>regex_filter</code> value, the <code>regex_filter</code> is set to <code>NULL</code>. See REGEXP_LIKE Condition for more information.

Parameter	Description
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: - logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation. The default value is 2 days. - logprefix: It accepts a string value that determines the bulk operation status table name prefix string. The operation type is the default value. For BULK_MOVE, the default logprefix value is MOVE. - priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and is completed sooner. It accepts the following values: - HIGH: Determines the number of parallel files handled using the database's ECPU count (OCPU count if your database uses OCPUs). - MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. - LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```

BEGIN
DBMS_CLOUD.BULK_MOVE (
    source_credential_name => 'OCI_CRED',
    source_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/o',
    target_location_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/o',
    format                 => JSON_OBJECT ('logretention' value 7,
'logprefix' value 'BULKMOVE')
);
END;
/

```

Note

An error is returned when the source and target URI point to the same Object Storage bucket or folder.

BULK_UPLOAD Procedure

This procedure copies files into Cloud Object Storage from an Autonomous AI Database directory. The overloaded form enables you to use the `operation_id` parameter.

Syntax

```
DBMS_CLOUD.BULK_UPLOAD (  
    credential_name IN VARCHAR2 DEFAULT NULL,  
    location_uri    IN VARCHAR2,  
    directory_name  IN VARCHAR2,  
    regex_filter    IN VARCHAR2 DEFAULT NULL,  
    format          IN CLOB      DEFAULT NULL  
);
```

```
DBMS_CLOUD.BULK_UPLOAD (  
    credential_name IN VARCHAR2 DEFAULT NULL,  
    location_uri    IN VARCHAR2,  
    directory_name  IN VARCHAR2,  
    regex_filter    IN VARCHAR2 DEFAULT NULL,  
    format          IN CLOB      DEFAULT NULL,  
    operation_id    OUT NUMBER  
);
```

Parameters

Parameter	Description
<code>credential_name</code>	The name of the credential to access the Cloud Object Storage. If you do not supply a <code>credential_name</code> value, the <code>credential_name</code> is set to NULL.
<code>location_uri</code>	Specifies URI, that points to an Object Storage location to upload files. This parameter is mandatory. The format of the URIs depends on the Cloud Object Storage service. See DBMS_CLOUD Package File Cloud Object Storage URI Formats for more information.

Parameter	Description
directory_name	The name of the directory on the Autonomous AI Database from where you upload files. This parameter is mandatory. Directory You can specify one directory and one or more file names or use a comma separated list of directories and file names. The format to specify a directory is: <code>'MY_DIR: filename.ext'</code>. By default the directory name <code>MY_DIR</code> is a database object and is case-insensitive. The file name is case sensitive. You can use wildcards to specify file names in a directory. The character <code>"**"</code> can be used as the wildcard for multiple characters, the character <code>"?"</code> can be used as the wildcard for a single character. For example: <code>'MY_DIR: *'</code> or <code>'MY_DIR: test?'</code> To specify multiple directories, use a comma separated list of directories: For example: <code>'MY_DIR1: *, MY_DIR2: test?'</code> Use double quotes to specify a case-sensitive directory name. For example: <code>'"my_dir1": *, "my_dir2": Test?'</code> To include a quote character, use two quotes. For example: <code>'MY_DIR: ' 'filename.ext'</code>. This specifies the filename starts with a quote (<code>'</code>).
regex_filter	Specifies the REGEX expression to filter files. The REGEX expression pattern must be compatible with REGEXP_LIKE operator. If you do not supply a regex_filter value, the regex_filter is set to NULL. See REGEXP_LIKE Condition for more information.
format	Specifies the additional configuration options for the file operation. These options are specified as a JSON string. The supported format options are: • logretention: It accepts an integer value that determines the duration in days for which the status table is retained for a bulk operation. The default value is 2 days. • logprefix: It accepts a string value that determines the bulk operation status table name prefix string. The operation type is the default value. For BULK_UPLOAD, the default logprefix value is UPLOAD. • priority: It accepts a string value that determines the number of file operations performed concurrently. An operation with a higher priority consumes more database resources and is completed sooner. It accepts the following values: – HIGH: Determines the number of parallel files handled using the database's ECPU count (OCPU count if your database uses OCPUs). – MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. – LOW: Process the files in serial order. The default value is MEDIUM. The maximum number of concurrent file operations is limited to 64. If you do not supply a format value, the format is set to NULL.

Parameter	Description
operation_id	Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view.

Example

```

BEGIN
DBMS_CLOUD.BULK_UPLOAD (
    credential_name => 'OCI_CRED',
    location_uri    => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o',
    directory_name  => 'BULK_TEST',
    format          => JSON_OBJECT ('logretention' value 5, 'logprefix'
value 'BULKUPLOAD')
);
END;
/

```

DBMS_CLOUD REST APIs

This section covers the DBMS_CLOUD REST APIs provided with Autonomous AI Database on Dedicated Exadata Infrastructure.

Prerequisites

As a developer, you can use DBMS_CLOUD procedures with Autonomous AI Databases deployed on Oracle Public Cloud, Multicloud or Exadata Cloud@Customer.

Depending on the deployment choice, the following prerequisites must be met to use the DBMS_CLOUD REST APIs with Amazon S3, Azure Blob Storage, and Google Cloud Storage service providers.

-
- [Oracle Public Cloud or Multicloud](#)
 - [Exadata Cloud@Customer](#)

Oracle Public Cloud or Multicloud

An outbound connectivity must have been configured using a NAT gateway, by your fleet administrator as described below:

- Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
- After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 1. Go to the **Subnet Details** page for the subnet.
 2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.

3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** NAT Gateway
 - **Target:** The name of the NAT gateway you just created in the VCNIf such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.
4. Return to the **Subnet Details** page for the subnet.
5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
6. In the side menu, under **Resources**, click **Egress Rules**.
7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** AllIf such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

DBMS_CLOUD REST APIs

This section covers the DBMS_CLOUD REST APIs provided with Autonomous AI Database.

REST API	Description
GET_RESPONSE_HEADERS Function	This function returns the HTTP response headers as JSON data in a JSON object in Autonomous AI Database.
GET_RESPONSE_TEXT Function	This function returns the HTTP response in TEXT format (VARCHAR2 or CLOB) in Autonomous AI Database. Usually, most Cloud REST APIs return JSON response in text format. This function is useful if you expect the HTTP response is in text format.
GET_API_RESULT_CACHE_SIZE Function	This function returns the configured result cache size.
SEND_REQUEST Function and Procedure	This function begins an HTTP request, gets the response, and ends the response in Autonomous AI Database. This function provides a workflow for sending a Cloud REST API request with arguments and a return response code and payload.
SET_API_RESULT_CACHE_SIZE Procedure	This procedure sets the maximum cache size for current session.

DBMS_CLOUD REST API Overview

When you use PL/SQL in your application and you need to call Cloud REST APIs you can use `DBMS_CLOUD.SEND_REQUEST` to send the REST API requests.

The `DBMS_CLOUD` REST API functions allow you to make HTTP requests using `DBMS_CLOUD.SEND_REQUEST` and obtain and save results. These functions provide a generic API that lets you call any REST API with the following supported cloud services:

- Oracle Cloud Infrastructure
See [API Reference and Endpoints](#) for information on Oracle Cloud Infrastructure REST APIs.
- Azure Cloud ¹
See [Azure REST API Reference](#) for information on Azure REST APIs.

DBMS_CLOUD REST API Constants

Describes the `DBMS_CLOUD` constants for making HTTP requests using `DBMS_CLOUD.SEND_REQUEST`.

`DBMS_CLOUD` supports GET, PUT, POST, HEAD and DELETE HTTP methods. The REST API method to be used for an HTTP request is typically documented in the Cloud REST API documentation.

Name	Type	Value
<code>METHOD_DELETE</code>	<code>VARCHAR2(6)</code>	<code>'DELETE'</code>
<code>METHOD_GET</code>	<code>VARCHAR2(3)</code>	<code>'GET'</code>
<code>METHOD_HEAD</code>	<code>VARCHAR2(4)</code>	<code>'HEAD'</code>
<code>METHOD_POST</code>	<code>VARCHAR2(4)</code>	<code>'POST'</code>
<code>METHOD_PUT</code>	<code>VARCHAR2(3)</code>	<code>'PUT'</code>

¹ Support for Azure Cloud REST API calls is limited to the domain "blob.windows.net".

DBMS_CLOUD REST API Results Cache

You can save DBMS_CLOUD REST API results when you set the cache parameter to true with DBMS_CLOUD.SEND_REQUEST. The SESSION_CLOUD_API_RESULTS view describes the columns you can use when REST API results are saved.

By default DBMS_CLOUD REST API calls do not save results for your session. In this case you use the DBMS_CLOUD.SEND_REQUEST function to return results.

When you use DBMS_CLOUD.SEND_REQUEST and set the cache parameter to TRUE, results are saved and you can view past results in the SESSION_CLOUD_API_RESULTS view. Saving and querying historical results of DBMS_CLOUD REST API requests can help you when you need to work with your previous results in your applications.

For example, to query recent DBMS_CLOUD REST API results, use the view SESSION_CLOUD_API_RESULTS:

```
SELECT timestamp FROM SESSION_CLOUD_API_RESULTS;
```

When you save DBMS_CLOUD REST API results with DBMS_CLOUD.SEND_REQUEST the saved data is only available within the same session (connection). After the session exits, the saved data is no longer available.

Use DBMS_CLOUD.GET_API_RESULT_CACHE_SIZE and DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE to view and set the DBMS_CLOUD REST API cache size, and to disable caching.

DBMS_CLOUD REST API Results cache_scope Parameter

When you save DBMS_CLOUD REST API results with DBMS_CLOUD.SEND_REQUEST, access to the results in SESSION_CLOUD_API_RESULTS is provided based on the value of cache_scope.

By default cache_scope is 'PRIVATE' and only the current user of the session can access the results. If you set the cache_scope to 'PUBLIC', then all session users can access the results. The default value for cache_scope specifies that each user can only see DBMS_CLOUD.SEND_REQUEST REST API results generated by the procedures they invoke with invoker's rights. When you invoke DBMS_CLOUD.SEND_REQUEST in a session, there are three possibilities that determines if the current user can see results in the cache, based on the cache_scope value:

- You directly execute DBMS_CLOUD.SEND_REQUEST as a top-level statement and the call to DBMS_CLOUD.SEND_REQUEST and the REST API results are saved with the same username. In this case you have access to all results with the default value, 'PRIVATE', set for cache_scope.
- You write a wrapper invoker's rights procedure and as the current user your call with DBMS_CLOUD.SEND_REQUEST calls the procedure and the REST API results are saved with the same username. In this case, and you have access to all results with the default value, 'PRIVATE', set for cache_scope.
- You write a wrapper definer's rights procedure and the procedure is owned by another user. When you call DBMS_CLOUD.SEND_REQUEST inside the procedure, the results are saved with the username of the procedure owner.

For this case, a different definer's rights user is invoking DBMS_CLOUD.SEND_REQUEST, and the REST API results are saved with that definer's procedure's owner. For this case, by default when cache_scope is 'PRIVATE', the invoker's session cannot see the results.

If the definer's procedure owner wants to make the results available to any invoking session user, then they must set `cache_scope` to 'PUBLIC' in the `DBMS_CLOUD.SEND_REQUEST`.

DBMS_CLOUD REST API SESSION_CLOUD_API_RESULTS View

You can save `DBMS_CLOUD REST API` results when you set the `cache` parameter to true with `DBMS_CLOUD.SEND_REQUEST`. The `SESSION_CLOUD_API_RESULTS` view describes the columns you can use when REST API results are saved.

The view `SESSION_CLOUD_API_RESULTS` is the view created if you cache results with `DBMS_CLOUD.SEND_REQUEST`. You can query historical results which belong to your user session. When the session ends, the data in the `SESSION_CLOUD_API_RESULTS` is purged.

Column	Description
URI	The <code>DBMS_CLOUD REST API</code> request URL
TIMESTAMP	The <code>DBMS_CLOUD REST API</code> response timestamp
CLOUD_TYPE	The <code>DBMS_CLOUD REST API</code> cloud type, such as Oracle Cloud Infrastructure and <code>AZURE_BLOB</code>
REQUEST_METHOD	The <code>DBMS_CLOUD REST API</code> request method, such as GET, PUT, HEAD
REQUEST_HEADERS	The <code>DBMS_CLOUD REST API</code> request headers
REQUEST_BODY_TEXT	The <code>DBMS_CLOUD REST API</code> request body in CLOB
RESPONSE_STATUS_CODE	The <code>DBMS_CLOUD REST API</code> response status code, such as 200 (OK), 404 (Not Found)
RESPONSE_HEADERS	The <code>DBMS_CLOUD REST API</code> response headers
RESPONSE_BODY_TEXT	The <code>DBMS_CLOUD REST API</code> response body in CLOB
SCOPE	The <code>cache_scope</code> set by <code>DBMS_CLOUD.SEND_REQUEST</code> . Valid values are PUBLIC or PRIVATE.

GET_RESPONSE_HEADERS Function

This function returns the HTTP response headers as JSON data in a JSON object.

Syntax

```
DBMS_CLOUD.GET_RESPONSE_HEADERS(  
    resp          IN DBMS_CLOUD_TYPES.resp)  
    RETURN JSON_OBJECT_T;
```

Parameters

Parameter	Description
resp	HTTP Response type returned from <code>DBMS_CLOUD.SEND_REQUEST</code> .

Exceptions

Exception	Error	Description
invalid_response	ORA-20025	Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_HELPERS.

GET_RESPONSE_TEXT Function

This function returns the HTTP response in TEXT format (VARCHAR2 or CLOB). Usually, most Cloud REST APIs return JSON response in text format. This function is useful if you expect the HTTP response is in text format.

Syntax

```
DBMS_CLOUD.GET_RESPONSE_TEXT(  
    resp          IN DBMS_CLOUD_TYPES.resp)  
    RETURN CLOB;
```

Parameters

Parameter	Description
resp	HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST.

Exceptions

Exception	Error	Description
invalid_response	ORA-20025	Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_TEXT.

GET_API_RESULT_CACHE_SIZE Function

This function returns the configured result cache size. The cache size value only applies for the current session.

Syntax

```
DBMS_CLOUD.GET_API_RESULT_CACHE_SIZE()  
    RETURN NUMBER;
```

SEND_REQUEST Function and Procedure

This function and procedure begins an HTTP request, gets the response, and ends the response. This function provides a workflow for sending a cloud REST API request with arguments and the function returns a response code and payload. If you use the procedure, you can view results and response details from the saved results with the SESSION_CLOUD_API_RESULTS view.

Syntax

```

DBMS_CLOUD.SEND_REQUEST(
    credential_name    IN VARCHAR2,
    uri                IN VARCHAR2,
    method              IN VARCHAR2,
    headers             IN CLOB DEFAULT NULL,
    async_request_url  IN VARCHAR2 DEFAULT NULL,
    wait_for_states    IN DBMS_CLOUD_TYPES.wait_for_states_t DEFAULT NULL,
    timeout             IN NUMBER DEFAULT 0,
    cache              IN PL/SQL BOOLEAN DEFAULT FALSE,
    cache_scope        IN VARCHAR2 DEFAULT 'PRIVATE',
    body               IN BLOB DEFAULT NULL)
RETURN DBMS_CLOUD_TYPES.resp;

```

```

DBMS_CLOUD.SEND_REQUEST(
    credential_name    IN VARCHAR2,
    uri                IN VARCHAR2,
    method              IN VARCHAR2,
    headers             IN CLOB DEFAULT NULL,
    async_request_url  IN VARCHAR2 DEFAULT NULL,
    wait_for_states    IN DBMS_CLOUD_TYPES.wait_for_states_t DEFAULT NULL,
    timeout             IN NUMBER DEFAULT 0,
    cache              IN PL/SQL BOOLEAN DEFAULT FALSE,
    cache_scope        IN VARCHAR2 DEFAULT 'PRIVATE',
    body               IN BLOB DEFAULT NULL);

```

Parameters

Parameter	Description
<code>credential_name</code>	The name of the credential for authenticating with the corresponding cloud native API.
<code>uri</code>	HTTP URI to make the request.
<code>method</code>	HTTP Request Method: GET, PUT, POST, HEAD, DELETE. Use the <code>DBMS_CLOUD</code> package constant to specify the method. See DBMS_CLOUD REST API Constants for more information.
<code>headers</code>	HTTP Request headers for the corresponding cloud native API in JSON format. The authentication headers are set automatically, only pass custom headers.
<code>async_request_url</code>	An asynchronous request URL. To obtain the URL select your request API from the list of APIs (see https://docs.cloud.oracle.com/en-us/iaas/api/). Then, navigate to find the API for your request in the left pane. For example, Database Services API → Autonomous AI Database → StopAutonomousDatabase. This page shows the API home (and shows the base endpoint). Then, append the base endpoint with the relative path obtained for your work request WorkRequest link.

Parameter	Description
wait_for_states	Wait for states is a status of type: DBMS_CLOUD_TYPES.wait_for_states_t. The following are valid values for expected states: 'ACTIVE', 'CANCELED', 'COMPLETED', 'DELETED', 'FAILED', 'SUCCEEDED'. Multiple states are allowed for wait_for_states. The default value for wait_for_states is to wait for any of the expected states: 'ACTIVE', 'CANCELED', 'COMPLETED', 'DELETED', 'FAILED', 'SUCCEEDED'.
timeout	Specifies the timeout, in seconds, for asynchronous requests with the parameters async_request_url and wait_for_states. Default value is 0. This indicates to wait for completion of the request without any timeout.
cache	If TRUE specifies the request should be cached in REST result API cache. The default value is FALSE, which means REST API requests are not cached.
cache_scope	Specifies whether everyone can have access to this request result cache. Valid values: "PRIVATE" and "PUBLIC". The default value is "PRIVATE".
body	HTTP Request Body for PUT and POST requests.

Exceptions

Exception	Error	Description
invalid_req_method	ORA-20023	Request method passed to DBMS_CLOUD.SEND_REQUEST is invalid.
invalid_req_header	ORA-20024	Request headers passed to DBMS_CLOUD.SEND_REQUEST are not in valid JSON format.

Usage Notes

- If you are using Oracle Cloud Infrastructure, you must use a Signing Key based credential value for the credential_name. See CREATE_CREDENTIAL Procedure for more information.
- The optional parameters async_request_url, wait_for_states, and timeout allow you to handle long running requests. Using this asynchronous form of send_request, the function waits for the completion status specified in wait_for_states before returning. With these parameters in the send request, you pass the expected return states in the wait_for_states parameter, and you use the async_request_url parameter to specify an associated work request, the request does not return immediately. Instead, the request probes the async_request_url until the return state is one of the expected states or the timeout is exceeded (timeout is optional). If no timeout is specified, the request waits until a state found in wait_for_states occurs.

SET_API_RESULT_CACHE_SIZE Procedure

This procedure sets the maximum cache size for current session. The cache size value only applies for the current session.

Syntax

```
DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE(  
    cache_size          IN NUMBER);
```

Parameters

Parameter	Description
cache_size	Set the maximum cache size to the specified value <code>cache_size</code> . If the new maximum cache size is smaller than the current cache size, older records are dropped until the number of rows is equal to the specified maximum cache size. The maximum value is 10000. If the cache size is set to 0, caching is disabled in the session. The default cache size is 10.

Exceptions

Exception	Error	Description
invalid API result cache size	ORA-20032	The minimum value is 0 and the maximum value is 10000. This exception is shown when the input value is less than 0 or is larger than 10000.

Example

```
EXEC DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE(101);
```

DBMS_CLOUD Package Format Options

The `format` argument in `DBMS_CLOUD` specifies the format of source files.

The two ways to specify the `format` argument are:

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

And:

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

Examples:

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

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

For example:

```
format => json_object('ignoremissingcolumns' value 'true', 'removequotes' value 'true',  
                    'dateformat' value 'YYYY-MM-DD-HH24-MI-SS', 'blankasnull'  
                    value 'true', 'logretention' value 7)
```

Note

For Avro, ORC, or Parquet format options, see DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet.

As noted in the **Format Option** column, a limited set of format options are valid with DBMS_CLOUD.COPY_COLLECTION or with DBMS_CLOUD.COPY_DATA when the format type is JSON.

Format Option	Description	Syntax
blankasnull	When set to true, loads fields consisting of spaces as null.	blankasnull : true Default value: False
characterset Valid with format JSON and COPY_DATA	Specifies the character set of source files	characterset: <i>string</i> Default value: Database character set
columnpath Only use with format JSON and COPY_DATA	Array of JSON path expressions that correspond to the fields that need to be extracted from the JSON records. Each of the JSON path expressions in the array should follow the rules described in <i>SQL/JSON Path Expressions</i> in Oracle Database 19c JSON Developer's Guide or Oracle Database 23ai JSON Developer's Guide . Only use with format JSON and DBMS_CLOUD.COPY_DATA.	JSON Array of json path expressions expressed in string format. For example: 'columnpath' value '["\$\$.WEATHER_STATION_ID", "\$\$.WEATHER_STATION_NAME"]'
compression Option valid with JSON data	Specifies the compression type of the source file. ZIP archiving format is not supported. Specifying the value auto checks for the compression types: gzip, zlib, std, bzip2.	compression: auto gzip zlib bzip2 Default value: Null value meaning no compression.
conversionerrors	If a row is rejected because of data type conversion errors, the related columns are stored as null or the row is rejected.	conversionerrors : reject_record store_null Default value: reject_record
dateformat	Specifies the date format in the source file. The format option AUTO searches for the following formats: J MM-DD-YYYYBC MM-DD-YYYY YYYYMMDD HHMISS YYMMDD HHMISS YYYY.DDD YYYY-MM-DD	dateformat : <i>string</i> Default value: Database date format
delimiter	Specifies the field delimiter. To use a special character as the delimiter, specify the HEX value of the ASCII code of the character. For example, the following specifies the TAB character as the delimiter: format => json_object('delimiter' value 'X'9')	delimiter : <i>character</i> Default value (pipe character)

Format Option	Description	Syntax
detectfieldorder	Specifies that the fields in the external data files are in a different order than the columns in the table. Detect the order of fields using the first row of each external data file and map it to the columns of the table. The field names in external data files are compared in case insensitive manner with the names of the table columns. This format option is applicable for the following procedures: - DBMS_CLOUD.COPY_DATA - DBMS_CLOUD.CREATE_EXTERNAL_TABLE - DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE - DBMS_CLOUD.CREATE_HYBRID_PART_TABLE Restrictions for detectfieldorder: - Field names in the data file must appear in the first record line and must not contain any white spaces between the field names. - The field delimiter in the field names record must be the same as the field delimiter for the data in the file. - Quoted field names are not supported. The field names in data files are compared, in case insensitive manner, with the names of the external table columns. - Embedded field delimiters are not allowed in the field names. - The number of columns in the table must match the number of fields in the data files. - This format option is not applicable for Bigdata or Oracle Data Pump formats, as those formats have precise column metadata information in the binary file format. The text formats, CSV, JSON, Parquet, or XML can benefit from this automatic field order detection when the first line contains the field names. See <i>FIELD NAMES</i> in Oracle Database 19c Utilities or Oracle Database 23ai Utilities and the description for ALL FILES for more information.	detectfieldorder: true Default value: false
enablelogs	The format option enablelogs is used with the following DBMS_CLOUD procedures: - COPY_DATA - COPY_COLLECTION - EXPORT_DATA enablelogs specifies a boolean value, when set to TRUE, logs are generated. When set to FALSE, logs are not generated. For example: <pre>format => JSON_OBJECT('enablelogs' value FALSE)</pre>	enablelogs: false Default value: true

Format Option	Description	Syntax
encryption	The format option encryption specifies the encryption and decryption options to export and import data to and from the Object Store. Use encryption to specify the following parameters to encrypt and decrypt: • <code>user_defined_function</code>: Specifies a fully qualified user defined function to decrypt or encrypt the specified BLOB (binary large object). It returns a decrypted or encrypted BLOB. <code>user_defined_function</code> is mutually exclusive with other parameters for encryption. For example, <code>ADMIN.DECRYPTON_CALLBACK</code>. • <code>type</code>: Specifies the <code>DBMS_CRYPT0</code> encryption algorithm to decrypt or encrypt. <code>type</code> accepts values in the <i>Block Cipher Algorithms + Block Cipher Chaining Modifiers + Block Cipher Padding Modifiers</i> format. Supported Block Cipher Algorithms are: – <code>DBMS_CRYPT0.ENCRYPT_AES256</code> Supported Block Cipher Chaining Modifiers are: – <code>DBMS_CRYPT0.CHAIN_CBC</code> – <code>DBMS_CRYPT0.CHAIN_CFB</code> – <code>DBMS_CRYPT0.CHAIN_ECB</code> – <code>DBMS_CRYPT0.CHAIN_OFB</code> Supported Block Cipher Padding Modifiers are: – <code>DBMS_CRYPT0.PAD_PKCS5</code> – <code>DBMS_CRYPT0.PAD_NONE</code> – <code>DBMS_CRYPT0.PAD_ZERO</code> – <code>DBMS_CRYPT0.PAD_ORCL</code> • <code>credential_name</code>: Specifies the credential used to store the encryption key. The Block Cipher Chaining Modifiers and Block Cipher Padding Modifiers values defaults to <code>DBMS_CRYPT0.CHAIN_CBC</code> and <code>DBMS_CRYPT0.PAD_PKCS5</code>, if you do not specify values for these parameters. The format option encryption is used with the following <code>DBMS_CLOUD</code> procedures: • Used to pass parameters to decrypt for these procedures: – <code>DBMS_CLOUD.COPY_DATA</code> – <code>DBMS_CLOUD.CREATE_EXTERNAL_TABLE</code> – <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> – <code>DBMS_CLOUD.CREATE_HYBRID_PART_TABLE</code> For <code>DBMS_CLOUD.CREATE_HYBRID_PART_TABLE</code> this option is only applicable to the Object Storage files. – <code>DBMS_CLOUD.COPY_COLLECTION</code> • Used to pass parameters to encrypt for these procedure:	<code>encryption:value</code> Where <i>value</i> is a JSON string that provides additional parameters for encryption: <code>type:value</code> Specifies the encryption type. <code>credential_name:value</code> Specifies the credential used to store the encryption key. <code>user_defined_function:value</code> Specifies a fully qualified user-defined function to decrypt or encrypt the specified BLOB (binary large object).

Format Option	Description	Syntax
	— DBMS_CLOUD.EXPORT_DATA For example: <pre>format => JSON_OBJECT('encryption' value json_object ('type' value DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC + DBMS_CRYPTO.PAD_PKCS5, 'credential_name' value 'ENCRYPTION_CRED'))</pre>	
endquote	Data can be enclosed between two delimiters, specified with quote and endquote. The quote and endquote characters are removed during loading when specified. For example: <pre>format => JSON_OBJECT('quote' value '(', 'endquote' value ')')</pre>	endquote:character Default value: Null, meaning no endquote.
escape	The character "\" is used as the escape character when specified.	escape : true Default value: false
ignoreblanklines Option valid with JSON data	Blank lines are ignored when set to true.	ignoreblanklines : true Default value: False
ignoremissingcolumns	If there are more columns in the field_list than there are in the source files, the extra columns are stored as null.	ignoremissingcolumns : true Default value False
jsonpath Only use with COPY_COLLECTION	JSON path to identify the document to load. This option is valid only for JSON collection data with DBMS_CLOUD.COPY_COLLECTION.	jsonpath: <i>string</i> Default value: Null
keyassignment Only use with COPY_COLLECTION	Specifies whether a new collection is created as a mongo-compatible collection or as a SODA collection. When the value is set to embedded_oid, a new collection is created as a mongo-compatible collection. By default this parameter is not set, meaning a new collection is created as a SODA collection.	keyassignment: embedded_oid Default: keyassignment is not set
keypath Only use with COPY_COLLECTION	Specifies an attribute in the data to be loaded as the '_id' value. If keypath is specified then you must also specify the keyassignment value as embedded_oid. Set the value to a path, for example, '\$.mykey', to pick the value of the path as '_id' value. This parameter is optional and is only valid for loading into mongo-compatible collections. If not specified, Oracle generates a 12-byte unique system ID and populates that as the '_id' attribute, if an '_id' attribute is not already present in the data being loaded.	keypath: <i>string</i> Default: keypath is not set. When keypath is set, the default <i>string</i> value is NULL.

Format Option	Description	Syntax
language	Specifies a language name (for example, FRENCH), from which locale-sensitive information can be derived.	language: <i>string</i> Default value: Null See <i>Locale Data</i> in Oracle Database 19c Database Globalization Support Guide or Oracle Database 23ai Database Globalization Support Guide for a listing of Oracle-supported languages.
logdir	Specifies a string value that determines the directory object name where the logfile_table or badfile_table files are saved. By default, the logdir is not case-sensitive, but the case is reserved when the specified value is enclosed in double-quotes. For example: format => JSON_OBJECT ('logdir' value 'test_log') The logdir format option specified in the above example saves the logfile_table or badfile_table files in the TEST_LOG directory object. format => JSON_OBJECT ('logdir' value '"test_log"') The logdir format option specified in the above example saves the logfile_table or badfile_table files in the test_log directory object.	logdir: <i>string</i> Default value: DATA_PUMP_DIR
logprefix	Specifies a string value that determines the prefix for the logfile_table and badfile_table files. The log table name format is: <i>logprefix\$operation_id</i> By default, the logprefix is in upper case, but the case is reserved when the specified value is enclosed in double-quotes. For example: format => JSON_OBJECT ('logprefix' value 'TEST') Log files then use the TEST prefix, such as: TEST\$2_LOG and TEST\$2_BAD.	logprefix: <i>string</i> Default value: COPY
logretention	Specifies a positive integer duration, in days, for which the logfile_table and badfile_table files are retained. Valid values: 0 to 99999 For example: format => JSON_OBJECT ('logretention' value 7)	logretention: <i>number</i> Default value: 2
maxdocsize This option is valid only with JSON data	Maximum size of JSON documents.	maxdocsize: <i>number</i> Default value: 1 Megabyte Maximum allowed value: 2 Gigabytes

Format Option	Description	Syntax
numericcharacters	Specifies the characters to use as the group separator and decimal character. <i>decimal_character</i>: The decimal separates the integer portion of a number from the decimal portion. <i>group_separator</i>: The group separator separates integer groups (that is, thousands, millions, billions, and so on).	numericcharacters: '<i>decimal_character</i> <i>group_separator</i>' Default value: ". , " See <i>NLS_NUMERIC_CHARACTERS</i> Oracle Database 19c Database Globalization Support Guide Oracle Database 23ai Database Globalization Support Guide for more information.
numberformat	Specifies the number format model. Number format models cause the number to be rounded to the specified number of significant digits. A number format model is composed of one or more number format elements. This is used in combination with <i>numericcharacters</i>.	numberformat: <i>number_format_model</i> Default value: is derived from the setting of the <i>NLS_TERRITORY</i> parameter See <i>Number Format Models</i> in Oracle Database 19c SQL Language Reference or Oracle Database 23ai SQL Language Reference for more information.

Format Option	Description	Syntax
partition_columns	The format option <code>partition_columns</code> is used with <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> to specify the column names and data types of partition columns when the partition columns are derived from the file path, depending on the type of data file, structured or unstructured: - When the <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> includes the <code>column_list</code> parameter and the data file is unstructured, such as with CSV text files, <code>partition_columns</code> does not include the data type. For example, use a format such as the following for this type of <code>partition_columns</code> specification: <pre>"partition_columns": ["state", "zipcode"]'</pre> The data type is not required because it is specified in the <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> <code>column_list</code> parameter. - When the <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> does not include the <code>column_list</code> parameter and the data files are structured, such as Avro, ORC, or Parquet files, the <code>partition_columns</code> option includes the data type. For example, the following shows a <code>partition_columns</code> specification: <pre>"partition_columns": [{"name": "country", "type": "varchar2(10)"}, {"name": "year", "type": "number"}, {"name": "month", "type": "varchar2(10)"}]</pre> If the data files are unstructured and the type sub-clause is specified with <code>partition_columns</code>, the type sub-clause is ignored. For object names that are not based on hive format, the order of the <code>partition_columns</code> specified columns must match the order as they appear in the object name in the <code>file_uri_list</code>.	
quote	Specifies the quote character for the fields, the quote characters are removed during loading when specified.	quote: <i>character</i> Default value: Null meaning no quote

Format Option	Description	Syntax
recorddelimiter	Specifies the record delimiter.	recorddelimiter: <i>character</i>
Option valid with JSON data	By default, DBMS_CLOUD tries to automatically find the correct newline character as the delimiter. It first searches the file for the Windows newline character "\r\n". If it finds the Windows newline character, this is used as the record delimiter for all files in the procedure. If a Windows newline character is not found, it searches for the UNIX/Linux newline character "\n" and if it finds one it uses "\n" as the record delimiter for all files in the procedure. Specify this argument explicitly if you want to override the default behavior, for example: <pre>format => json_object('recorddelimiter' VALUE '''\r\n''')</pre> To indicate that there is no record delimiter you can specify a recorddelimiter that does not occur in the input file. For example, to indicate that there is no delimiter, specify the control character 0x01 (SOH) as a value for the recorddelimiter and set the recorddelimiter value to "0x'01'" (this character does not occur in JSON text). For example: <pre>format => '{"recorddelimiter" : "0x'01'"}'</pre> The recorddelimiter is set once per procedure call. If you are using the default value, detected newline, then all files use the same record delimiter, if one is detected.	Default value: detected newline

Format Option	Description	Syntax
regexuri	The format option <code>regexuri</code> is used with the following DBMS_CLOUD procedures: - COPY_COLLECTION - COPY_DATA - CREATE_EXTERNAL_TABLE - CREATE_EXTERNAL_PART_TABLE - CREATE_HYBRID_PART_TABLE When the value of <code>regexuri</code> is set to TRUE, you can use wildcards as well as regular expressions in the file names in Cloud source file URIs. The characters "*" and "?" are considered wildcard characters when the <code>regexuri</code> parameter is set to FALSE. When the <code>regexuri</code> parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. Regular expression patterns are not supported for directory names. For external tables, this option is only supported with the tables that are created on a file in the Object Storage. For example: <pre>format => JSON_OBJECT('regexuri' value TRUE)</pre> See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 23ai SQL Language Reference for more information on REGEXP_LIKE condition.	<code>regexuri: True</code> Default value : False
rejectlimit	The operation will error out after specified number of rows are rejected.	<code>rejectlimit: number</code> Default value: 0
removequotes	Removes any quotes that are around any field in the source file.	<code>removequotes: true</code> Default value: False
skipheaders	Specifies how many rows should be skipped from the start of the file.	<code>skipheaders: number</code> Default value: 0 if not specified, 1 if specified without a value
territory	Specifies a territory name to further determine input data characteristics.	<code>territory: string</code> Default value: Null See <i>Locale Data</i> in Oracle Database 19c Database Globalization Support Guide or Oracle Database 23ai Database Globalization Support Guide for a listing of Oracle-supported languages.
timestampformat	Specifies the timestamp format in the source file. The format option AUTO searches for the following formats: <pre>YYYY-MM-DD HH:MI:SS.FF YYYY-MM-DD HH:MI:SS.FF3 YYYY-MM-DD HH24:MI:SS.FF3 MM/DD/YYYY HH:MI:SS.FF3</pre>	<code>timestampformat : string</code> Default value: Database timestamp format The string can contain wildcard characters such as "\$".

Format Option	Description	Syntax
timestamp_tzformat	Specifies the timestamp with local timezone format in the source file. The format option AUTO searches for the following formats: DD Mon YYYY HH:MI:SS.FF TZR MM/DD/YYYY HH:MI:SS.FF TZR YYYY-MM-DD HH:MI:SS+/-TZR YYYY-MM-DD HH:MI:SS.FF3 DD.MM.YYYY HH:MI:SS TZR	timestamp_tzformat : <i>string</i> Default value: Database timestamp with local timezone format
timestamp_tzformat	Specifies the timestamp with timezone format in the source file. The format option AUTO searches for the following formats: DD Mon YYYY HH:MI:SS.FF TZR MM/DD/YYYY HH:MI:SS.FF TZR YYYY-MM-DD HH:MI:SS+/-TZR YYYY-MM-DD HH:MI:SS.FF3 DD.MM.YYYY HH:MI:SS TZR	timestamp_tzformat: <i>string</i> Default value: Database timestamp with timezone format
trimspaces	Specifies how the leading and trailing spaces of the fields are trimmed. See the description of <i>trim_spec</i> in Oracle Database 19c Utilities or Oracle Database 23ai Utilities .	trimspaces: rtrim ltrim notrim ltrim ldrtrim Default value: notrim
truncatecol	If the data in the file is too long for a field, then this option will truncate the value of the field rather than reject the row.	truncatecol:true Default value: False

Format Option	Description	Syntax
type	Specifies the source file type. See the description of CSV in Oracle Database 19c Utilities or Oracle Database 23ai Utilities. If the type is datapump, then the only other valid format option is rejectlimit. If the type is datapump, then the only Object Stores supported are Oracle Cloud Infrastructure Object Storage and Oracle Cloud Infrastructure Object Storage Classic. See DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet for type values avro, orc, or parquet. For JSON data with DBMS_CLOUD.COPY_COLLECTION type has two valid values: json (default) and ejson. For DBMS_CLOUD.COPY_COLLECTION these values both specify that the input is JSON data. The value ejson causes extended objects in the textual JSON input data to be translated to scalar JSON values in the native binary JSON collection. The value json does not perform this transformation and all objects in the input data are converted to binary JSON format. For JSON data with DBMS_CLOUD.COPY_DATA type has one valid value: json. This value specifies that the input is JSON data. • See COPY_COLLECTION Procedure for information on loading JSON objects. • See Textual JSON Objects that Represent Extended Scalar Values for information about extended JSON objects, which are interpreted when parameter type has value ejson.	type: csv csv with embedded csv without embedded avro datapump orc parquet csv is the same as csv without embedded. Default value: Null For JSON data there are two valid type values for use with DBMS_CLOUD.COPY_COLLECTION: json ejson In this case the default value is json. For JSON data with DBMS_CLOUD.COPY_DATA, only json is valid.
unpackarrays Only use with COPY_COLLECTION	When set to true, if a loaded document is an array, then the contents of the array are loaded as documents rather than the array itself. This only applies to the top-level array. When set to true, the entire array is inserted as a single document. This option is valid only for JSON collection data with DBMS_CLOUD.COPY_COLLECTION.	unpackarrays: true Default value: False

Note

Not all DBMS_CLOUD procedures support all of these types.

DBMS_CLOUD Package Format Options for EXPORT_DATA

Describes the valid format parameter options for DBMS_CLOUD.EXPORT_DATA with text file formats, CSV, JSON, or XML.

These are the valid format parameters for use with DBMS_CLOUD.EXPORT_DATA. You specify text file output when you use the format type option and the value is one of: csv, json, parquet, or xml.

The two ways to specify the format argument are:

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

And:

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

Examples:

```
format => json_object('type' VALUE 'json')
```

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

For example:

```
format => json_object('compression' value 'gzip', 'type' value 'json')
```

This table covers the format options for DBMS_CLOUD.EXPORT_DATA when the format parameter type option is one of: CSV, JSON, Parquet, or XML. For other procedures and other output types, see DBMS_CLOUD Package Format Options for the list of format options.

Format Option	Description	Syntax
compression	Specifies the compression type of the source file. Note: ZIP archiving format is not supported. When the format type is csv, json, or xml, the default compression is Null, meaning no compression. When the format type is parquet, the default compression is snappy.	When the type is:csv json xml compression: gzip Default value: Null value meaning no compression. When the type is parquet compression: gzip snappy Default value: snappy
delimiter	Specifies a custom field delimiter. <pre>format => json_object('delimiter' value ' ')</pre> The delimiter value cannot be an ASCII code or an escape character.	delimiter: <i>character</i> Default value, (comma)
Note This option only applies with csv type.		
endquote	Specifies that fields can be enclosed between two delimiters, with quote and endquote. If endquote is not specified, then the quote character will be used by default as the endquote character. For example: <pre>format => JSON_OBJECT('quote' value '(', 'endquote' value ')')</pre>	endquote: <i>character</i> Default value: Null, meaning no endquote.
Note This option only applies with csv type.		

Format Option	Description	Syntax
escape	Specifies the occurrence of quote character in the field value using "\"" character.	escape : true Default value: false

 **Note**
This option only applies with csv type.

Format Option	Description	Syntax
encryption	The format option encryption specifies the encryption and decryption options to export and import data to and from the Object Store. Use encryption to specify the following parameters to encrypt and decrypt: • <code>user_defined_function</code>: Specifies a fully qualified user-defined function to decrypt or encrypt the specified BLOB (binary large object). It returns a decrypted or encrypted BLOB. This parameter is mutually exclusive with other parameters for encryption. For example, <code>ADMIN.DECRYPTON_CALLBACK</code>. • <code>type</code>: Specifies the built-in encryption algorithm to decrypt or encrypt. <code>user_defined_function</code> and <code>type</code> are mutually exclusive. <code>type</code> accepts values in the <i>Block Cipher Algorithms + Block Cipher Chaining Modifiers + Block Cipher Padding Modifiers</i> format. Supported Block Cipher Algorithms are: – <code>DBMS_CRYPTO.ENCRYPT_AES256</code> Supported Block Cipher Chaining Modifiers are: – <code>DBMS_CRYPTO.CHAIN_CBC</code> – <code>DBMS_CRYPTO.CHAIN_CFB</code> – <code>DBMS_CRYPTO.CHAIN_ECB</code> – <code>DBMS_CRYPTO.CHAIN_OFB</code> Supported Block Cipher Padding Modifiers are: – <code>DBMS_CRYPTO.PAD_PKCS5</code> – <code>DBMS_CRYPTO.PAD_NONE</code> – <code>DBMS_CRYPTO.PAD_ZERO</code> – <code>DBMS_CRYPTO.PAD_ORCL</code> • <code>credential_name</code>: Specifies the credential used to store the encryption key. The Block Cipher Chaining Modifiers and Block Cipher Padding Modifiers values defaults to <code>DBMS_CRYPTO.CHAIN_CBC</code> and <code>DBMS_CRYPTO.PAD_PKCS5</code>, if you do not specify values for these parameters. The format option encryption is used with the following <code>DBMS_CLOUD</code> procedures: • Used to pass parameters to decrypt for these procedures: – <code>DBMS_CLOUD.COPY_DATA</code> – <code>DBMS_CLOUD.CREATE_EXTERNAL_TABLE</code> – <code>DBMS_CLOUD.CREATE_HYBRID_TABLE</code> – <code>DBMS_CLOUD.COPY_COLLECTION</code> • Used to pass parameters to encrypt for these procedure: – <code>DBMS_CLOUD.EXPORT_DATA</code> For example: <pre>format => JSON_OBJECT('encryption' value json_object ('type' value DBMS_CRYPTO.ENCRYPT_AES256 + DBMS_CRYPTO.CHAIN_CBC +</pre>	<pre>encryption: value</pre> Where <i>value</i> is a JSON string that provides additional parameters for encryption: <pre>type: value</pre> Specifies the encryption type. <pre>credential_name: value</pre> Specifies the credential used to store the encryption key. <pre>user_defined_function: value</pre> Specifies a fully qualified user-defined function to decrypt or encrypt the specified BLOB (binary large object).

Format Option	Description	Syntax
header	DBMS_CLOUD.PAD_PKCS5, 'credential_name' value 'ENCRYPTION_CRED')) Writes column names as the first line in output files of csv type. The header option can accept a boolean or a string value. The valid values are: • false: Skips the header row. • true: Includes the header row. The column names are based on the SELECT statement in the query parameter. You must specify column aliases in the SELECT statement when using virtual columns or expressions. • <i>String to define custom header names</i>: Enables you to define header rows with custom names. The number of columns and delimiters in the string value must match the number of columns and delimiters in the SELECT statement. The default delimiter is comma (.). For example: <pre>format => JSON_OBJECT('type' value 'csv', 'delimiter' value ' ', 'compression' value 'gzip', 'header' value true)</pre>	header: true false <i>String to define custom header names</i> Default value: false
fileextension	Custom file extension to override the default choice for the format type. This applies to text formats with DBMS_CLOUD.EXPORT_DATA: CSV, JSON, Parquet, or XML. If the specified string does not start with period (dot), then a dot is automatically inserted before the file extension in the final file name. If no file extension is desired, use the value: fileextension = 'none'	Valid values: Any file extension. Default value: Depends on the format type option: • CSV format: .csv • JSON format: .json • PARQUET format: .parquet • XML format: .xml
maxfilesize	Number in bytes for maximum size of output generated. This applies to text based formats for exporting data with DBMS_CLOUD.EXPORT_DATA when the format type option is set to csv, json, or xml. Note: This option is not valid when the format type option is parquet.	Minimum value: 10485760 (10 MB) Maximum value: 1 GB Default value: 10485760 (10 MB)

Note

This option only applies with csv type.

Format Option	Description	Syntax
quote	In CSV format, fields can be enclosed between two delimiters. Specify the delimiters with quote and endquote. If endquote is not specified, then the quote character will be used by default as the endquote character.	quote: <i>character</i> Default value: Null meaning do not enclose fields with quotes.
	Note This option only applies with csv type.	
trimspace	Specifies how the leading and trailing spaces of the fields are trimmed for CSV format. Trim spaces is applied before quoting the field, if the quote parameter is specified. See the description of <i>trim_spec</i> in Oracle Database 19c Utilities or Oracle Database 26ai Utilities.	trimspace: rtrim ltrim notrim lrtrim ldrtrim Default value: notrim
	Note This option only applies with csv type.	
type	Specifies the output file type. csv: specifies Character Separated Values (CSV) format that allows you to export query results as a set of column values separated by any custom character. json: specifies to export the query results as JSON files. parquet: specifies to export the query results as Parquet files. xml: specifies to export query results as rows of valid XML documents. Each row is encapsulated in a root XML tag of <RECORD> </RECORD>. The query result is automatically transformed into XML format using XMLFOREST SQL function. Use Column Aliases to customize the XML tag names for columns. See <i>access_parameters Clause</i> in Oracle Database 19c Utilities or Oracle Database 26ai Utilities for more information.	type: csv datapump json parquet xml

DBMS_CLOUD Avro, ORC, and Parquet Support

This section covers the DBMS_CLOUD Avro, ORC, and Parquet support provided with Autonomous AI Database.

DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet

The format argument in DBMS_CLOUD specifies the format of source files.

The two ways to specify the format argument are:

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

And:

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

Examples:

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

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

For example:

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

Format Option	Description	Syntax
regexuri	When the value of regexuri is set to TRUE, you can use wildcards as well as regular expressions in the file names in Cloud source file URIs. The characters "*" and "?" are considered wildcard characters when the regexuri parameter is set to FALSE. When the regexuri parameter is set to TRUE the characters "*" and "?" are part of the specified regular expression pattern. Regular expression patterns are only supported for the file name or subfolder path in your URIs and the pattern matching is identical to that performed by the REGEXP_LIKE function. Regular expression patterns are not supported for directory names. For external tables, this option is only supported with the tables that are created on a file in the Object Storage. For example: <pre>format => JSON_OBJECT('regexuri' value TRUE)</pre> See <i>REGEXP_LIKE Condition</i> in Oracle Database 19c SQL Language Reference or Oracle Database 26ai SQL Language Reference for more information on REGEXP_LIKE condition.	regexuri: True Default value:False
type	Specifies the file type.	type : avro orc parquet

Format Option	Description	Syntax
schema	When schema is set to <code>first</code> or <code>all</code>, the external table columns and data types are automatically derived from the Avro, ORC, or Parquet file metadata. The column names will match those found in Avro, ORC, or Parquet. The data types are converted from Avro, ORC, or Parquet data types to Oracle data types. All columns are added to the table. The value <code>first</code> specifies to use the metadata from the first Avro, ORC, or Parquet file in the <code>file_uri_list</code> to auto generate the columns and their data types. Use <code>first</code> if all of the files have the same schema. The value <code>all</code> specifies to use the metadata from all Avro, ORC, or Parquet files in the <code>file_uri_list</code> to auto generate the columns and their data types. Use <code>all</code> (slower) if the files may have different schemas. Default: If <code>column_list</code> is specified, then the schema value, if specified is ignored. If <code>column_list</code> is not specified then the schema default value is <code>first</code>. Note: For Avro, ORC, or Parquet format files the schema format option is not available and the <code>column_list</code> parameter must be specified for partitioned external tables using the <code>DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE</code> procedure.	<code>schema : first all</code>

DBMS_CLOUD Package Avro to Oracle Data Type Mapping

Describes the mapping of Avro data types to Oracle data types.

Note

Complex types, such as maps, arrays, and structs are supported starting with Oracle Database 19c. See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using Avro complex types.

Avro Type	Oracle Type
INT	NUMBER(10)
LONG	NUMBER(19)
BOOL	NUMBER(1)
UTF8 BYTE_ARRAY	RAW(2000)
FLT	BINARY_FLOAT
DBL	BINARY_DOUBLE
DECIMAL(p)	NUMBER(p)
DECIMAL(p,s)	NUMBER(p,s)
DATE	DATE
STRING	VARCHAR2
TIME_MILLIS	VARCHAR2(20 BYTE)
TIME_MICROS	VARCHAR2(20 BYTE)
TIMESTAMP_MILLIS	TIMESTAMP(3)

Avro Type	Oracle Type
TIMESTAMP_MICROS	TIMESTAMP(6)
ENUM	VARCHAR2(<i>n</i>) Where: " <i>n</i> " is the actual maximum length of the AVRO ENUM's possible values
DURATION	RAW(2000)
FIXED	RAW(2000)
NULL	VARCHAR2(1) BYTE

See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using Avro complex types.

DBMS_CLOUD Package ORC to Oracle Data Type Mapping

Describes the mapping of ORC data types to Oracle data types.

See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using ORC complex types.

ORC Type	Oracle Type	More Information
array	VARCHAR2(<i>n</i>) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
bigint (64 bit)	NUMBER(19)	
binary	BLOB	
boolean (1 bit)	NUMBER(1)	
char	CHAR(<i>n</i>)	
date	DATE	
double	BINARY_DOUBLE	
float	BINARY_FLOAT	
int (32 bit)	NUMBER(10)	
list	VARCHAR2(<i>n</i>) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
map	VARCHAR2(<i>n</i>) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
smallint (16 bit)	NUMBER(5)	
string	VARCHAR2(4000)	
struct	VARCHAR2(<i>n</i>) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
timestamp	TIMESTAMP	
tinyint (8 bit)	NUMBER(3)	
union	VARCHAR2(<i>n</i>) JSON format	DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types
varchar	VARCHAR2(<i>n</i>)	

DBMS_CLOUD Package Parquet to Oracle Data Type Mapping

Describes the mapping of Parquet data types to Oracle data types.

Note

Complex types, such as maps, arrays, and structs are supported starting with Oracle Database 19c. See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using Parquet complex types.

Parquet Type	Oracle Type
UINT_64	NUMBER(20)
INT_64	NUMBER(19)
UINT_32	NUMBER(10)
INT_32	NUMBER(10)
UINT_16	NUMBER(5)
INT_16	NUMBER(5)
UINT_8	NUMBER(3)
INT_8	NUMBER(3)
BOOL	NUMBER(1)
UTF8 BYTE_ARRAY	VARCHAR2(4000 BYTE)
FLT	BINARY_FLOAT
DBL	BINARY_DOUBLE
DECIMAL(p)	NUMBER(p)
DECIMAL(p,s)	NUMBER(p,s)
DATE	DATE
STRING	VARCHAR2(4000)
TIME_MILLIS	VARCHAR2(20 BYTE)
TIME_MILLIS_UTC	VARCHAR2(20 BYTE)
TIME_MICROS	VARCHAR2(20 BYTE)
TIME_MICROS_UTC	VARCHAR2(20 BYTE)
TIMESTAMP_MILLIS	TIMESTAMP(3)
TIMESTAMP_MILLIS_UTC	TIMESTAMP(3)
TIMESTAMP_MICROS	TIMESTAMP(6)
TIMESTAMP_MICROS_UTC	TIMESTAMP(6)
TIMESTAMP_NANOS	TIMESTAMP(9)

See [DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types](#) for information on using Parquet complex types.

DBMS_CLOUD Package Oracle Data Type to Parquet Mapping

Describes the mapping of Oracle data types to Parquet data types.

Oracle Type	Parquet Type
BINARY_DOUBLE	DBL

Oracle Type	Parquet Type
BINARY_FLOAT	FLT
DATE	DATE
NUMBER(p,s)	DECIMAL(p,s)
NUMBER(p)	DECIMAL(p)
TIMESTAMP(3)	TIMESTAMP_MILLIS
TIMESTAMP(3)	TIMESTAMP_MILLIS_UTC
TIMESTAMP(6)	TIMESTAMP_MICROS
TIMESTAMP(6)	TIMESTAMP_MICROS_UTC
TIMESTAMP(9)	TIMESTAMP_NANOS
VARCHAR2(4000)	STRING

NLS Session Parameters

The NLS session parameters `NLS_DATE_FORMAT`, `NLS_TIMESTAMP_FORMAT`, `NLS_TIMESTAMP_TZ_FORMAT` and `NLS_NUMERIC_CHARACTERS` define how the date, timestamp, timestamp with time zone format, and radix separator for timestamp with decimal marker should be shown when a table with those column types are queried.

In addition, when you export data using `DBMS_CLOUD.EXPORT_DATA` and specify Parquet output, Autonomous AI Database reads the values of these parameters from the `NLS_SESSION_PARAMETERS` table. Autonomous AI Database uses these values to convert the Oracle data types `DATE` or `TIMESTAMP` to Parquet types.

The `NLS_SESSION_PARAMETERS` parameters support an `RR` format mask (two character year specification).

The `RR` format mask for the year is not supported for these parameters when you export data to Parquet with `DBMS_CLOUD.EXPORT_DATA`. An application error is raised if you attempt to export to parquet and the `NLS_SESSION_PARAMETERS` are set to use the `RR` format mask (the default value for the `RR` format depends on the value of the `NLS_TERRITORY` parameter).

When one of the parameters `NLS_DATE_FORMAT`, `NLS_TIMESTAMP_FORMAT` or `NLS_TIMESTAMP_TZ_FORMAT` uses the `RR` format mask, you must change the format value to supported value to export data to Parquet with `DBMS_CLOUD.EXPORT_DATA`. For example:

```
ALTER SESSION SET NLS_DATE_FORMAT = "MM/DD/YYYY";
ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH:MI:SS.FF';
ALTER SESSION SET NLS_TIMESTAMP_TZ_FORMAT='YYYY-MM-DD HH:MI:SS.FF TZH:TZM';
```

After you change the value, you can verify the change by querying the `NLS_SESSION_PARAMETERS` view:

```
SELECT value FROM NLS_SESSION_PARAMETERS
       WHERE parameter IN
('NLS_DATE_FORMAT', 'NLS_TIMESTAMP_FORMAT', 'NLS_TIMESTAMP_TZ_FORMAT');
```

If `NLS_DATE_FORMAT` is set, it applies to the columns with `DATE` datatype. If `NLS_TIMESTAMP_FORMAT` is set, it applies to the columns with `TIMESTAMP` datatype. If `NLS_TIMESTAMP_TZ_FORMAT` is set, it applies to the columns with `TIMESTAMP WITH TIME ZONE` datatype.

See *Date and Time Parameters* in Oracle Database 19c Database Globalization Support Guide or [Oracle Database 26ai Database Globalization Support Guide](#) and *NLS Data Dictionary Views* in Oracle Database 19c Database Globalization Support Guide or [Oracle Database 26ai Database Globalization Support Guide](#) for more information.

DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types

Describes the mapping of Avro, ORC, and Parquet complex data types to Oracle data types.

Autonomous AI Database supports complex data types, including the following complex types:

- struct
- list
- map
- union
- array

When you specify a source file type of Avro, ORC, or Parquet and the source file includes complex columns, Autonomous AI Database queries return JSON for the complex columns. This simplifies processing of query results; you can use Oracle's powerful JSON parsing features consistently across the file types and data types. The following table shows the format for the complex types in Autonomous AI Database:

Note

The complex fields map to VARCHAR2 columns and VARCHAR2 size limits apply.

Type	Parquet	ORC	Avro	Oracle
List: sequence of values	List	List	Array	VARCHAR2 (JSON format)
Map: list of objects with single key	Map	Map	Map	VARCHAR2 (JSON format)
Union: values of different type	Not Available	Union	Union	VARCHAR2 (JSON format)
Object: zero or more key-value pairs	Struct	Struct	Record	VARCHAR2 (JSON format)

If your ORC, Parquet, or Avro source files contain complex types, then you can query the JSON output for these common complex types. For example, the following shows an ORC file, `movie-info.orc`, with a complex type (the same complex type handling applies for Parquet and Avro source files).

Consider the `movie-info.orc` file with the following schema:

```
id      int
original_title string
overview      string
poster_path  string
release_date  string
vote_count   int
runtime      int
```

```
popularity    double
genres        array<struct<id:int,name:string>
```

Notice that each movie is categorized by multiple genres using an array of genres. The genres array is an array of structs and each item has an id (int) and a name (string). The genres array is considered a complex type. You can create a table over this ORC file using DBMS_CLOUD.CREATE_EXTERNAL_TABLE as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
    table_name => 'movie_info',
    credential_name => 'OBJ_STORE_CRED',
    file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
mytenancy/b/movies/o/movie-info.orc',
    format => '{"type": "orc", "schema": "first"}');
END;
/
```

When you create the external table the database automatically generates the columns based on the schema in the ORC file (if you are using Avro or Parquet, the same applies). For this example, the DBMS_CLOUD.CREATE_EXTERNAL_TABLE creates a table in your database as follows:

```
CREATE TABLE "ADMIN"."MOVIE_INFO"
( "ID"
  NUMBER(10,0),
  "ORIGINAL_TITLE"  VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "OVERVIEW"        VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "POSTER_PATH"     VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "RELEASE_DATE"    VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
  "VOTE_COUNT"      NUMBER(10,0),
  "RUNTIME"         NUMBER(10,0),
  "POPULARITY"      BINARY_DOUBLE,
  "GENRES"          VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP"
) DEFAULT COLLATION "USING_NLS_COMP"
ORGANIZATION EXTERNAL
( TYPE          ORACLE_BIGDATA
  DEFAULT DIRECTORY "DATA_PUMP_DIR"
  ACCESS PARAMETERS
  ( com.oracle.bigdata.credential.name=OBJ_STORE_CRED
    com.oracle.bigdata.fileformat=ORC
  )
  LOCATION
  (
    'https://objectstorage.us-phoenix-1.oraclecloud.com/n/mytenancy/b/
movies/o/movie-info.orc'
  )
)
REJECT LIMIT UNLIMITED
PARALLEL;
)
```

Now you can query the movie data:

```
SELECT original_title, release_date, genres
FROM movie_info
WHERE release_date > '2000'
ORDER BY original_title;
```

This produces the following output:

```
original_title      release_date  genres
(500) Days of Summer  2009        [{"id":3,"name":"Drama"},
{"id":6,"name":"Comedy"}, {"id":17,"name":"Horror"}, {"id":19,"name":"Western"},
{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
10,000 BC           2008        [{"id":6,"name":"Comedy"}]
11:14              2003        [{"id":9,"name":"Thriller"},
{"id":14,"name":"Family"}]
127 Hours           2010        [{"id":6,"name":"Comedy"},
{"id":3,"name":"Drama"}]
13 Going on 30      2004        [{"id":6,"name":"Comedy"},
{"id":3,"name":"Drama"}, {"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
1408                2007        [{"id":45,"name":"Sci-Fi"},
{"id":6,"name":"Comedy"}, {"id":17,"name":"Horror"}, {"id":6,"name":"Comedy"},
{"id":18,"name":"War"}]
```

Notice that the complex type genres is returned as a JSON array.

To make the JSON data more useful, you can transform the column using Oracle's JSON functions. For example, you can use the JSON "." notation as well as the more powerful transform functions such as JSON_TABLE.

See *Simple Dot-Notation Access to JSON Data* in Oracle Database 19c JSON Developer's Guide or [Oracle Database 26ai JSON Developer's Guide](#) for information on "." notation.

See *SQL/JSON Function JSON_TABLE* in Oracle Database 19c JSON Developer's Guide or [Oracle Database 26ai JSON Developer's Guide](#) for information on JSON_TABLE.

The following example shows a query on the table that takes each value of the array and turns the value into a row in the result set:

```
SELECT original_title, release_date, m.genre_name, genres
FROM movie_info mi,
JSON_TABLE(mi.genres, '$.name[*]'
COLUMNS (genre_name VARCHAR2(25) PATH
'$')
) AS m
WHERE rownum < 10;
```

The JSON_TABLE creates a row for each value of the array, think outer join, and the struct is parsed to extract the name of the genre. This produces the following output:

```
original_title      release_date  genre_name  genres
(500) Days of Summer  2009        Drama      [{"id":3,"name":"Drama"}, {"id":6,"name":"Comedy"},
```

```

{"id":17,"name":"Horror"}, {"id":19,"name":"Western"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
(500) Days of Summer          2009          Comedy
[{"id":3,"name":"Drama"}, {"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"}, {"id":19,"name":"Western"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
(500) Days of Summer          2009          Horror
[{"id":3,"name":"Drama"}, {"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"}, {"id":19,"name":"Western"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
(500) Days of Summer          2009          Western
[{"id":3,"name":"Drama"}, {"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"}, {"id":19,"name":"Western"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
(500) Days of Summer          2009          War
[{"id":3,"name":"Drama"}, {"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"}, {"id":19,"name":"Western"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
(500) Days of Summer          2009          Romance
[{"id":3,"name":"Drama"}, {"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"}, {"id":19,"name":"Western"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
10,000 BC                    2008          Comedy
[{"id":6,"name":"Comedy"}]
11:14                        2003          Family
[{"id":9,"name":"Thriller"}, {"id":14,"name":"Family"}]
11:14                        2003          Thriller
[{"id":9,"name":"Thriller"}, {"id":14,"name":"Family"}]
127 Hours                    2010          Comedy
[{"id":6,"name":"Comedy"}, {"id":3,"name":"Drama"}]
127 Hours                    2010          Drama
[{"id":6,"name":"Comedy"}, {"id":3,"name":"Drama"}]
13 Going on 30               2004          Romance
[{"id":6,"name":"Comedy"}, {"id":3,"name":"Drama"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
13 Going on 30               2004          Comedy
[{"id":6,"name":"Comedy"}, {"id":3,"name":"Drama"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]
13 Going on 30               2004          War
[{"id":6,"name":"Comedy"}, {"id":3,"name":"Drama"},

{"id":18,"name":"War"}, {"id":15,"name":"Romance"}]

```

```
13 Going on 30                                2004                                Drama
[{"id":6, "name":"Comedy"}, {"id":3, "name":"Drama"},

{"id":18, "name":"War"}, {"id":15, "name":"Romance"}]
```

DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping

Describes rules for how Avro, ORC, and Parquet column names are converted to Oracle column names.

The following are supported for Avro, ORC, and Parquet column names, but may require use of double quotes for Oracle SQL references in external tables. Thus, for ease of use and to avoid having to use double quotes when referencing column names, if possible do not use the following in Avro, ORC, and Parquet column names:

- Embedded blanks
- Leading numbers
- Leading underscores
- Oracle SQL reserved words

The following table shows various types of Avro, ORC, and Parquet column names, and rules for using the column names in Oracle column names in external tables.

Avro, ORC, or Parquet Name	CREATE TABLE Name	Oracle CATALOG	Valid SQL	Notes
part, Part, or PART	part, Part, PART	PART	select part select Part select paRt select PART	Oracle implicitly uppercases unquoted column names
Ord No	"Ord No"	Ord No	select "Ord No"	Double quotes are required when there are embedded blanks, which also preserves the character case
__index_key__	"__index_key__"	__index_key__	select "__index_key__"	Double quotes are required when there is a leading underscore, which also preserves the character case
6Way	"6Way"	6Way	select "6Way"	Double quotes are required when there is a leading numeric digit, which also preserves the character case

Avro, ORC, or Parquet Name	CREATE TABLE Name	Oracle CATALOG	Valid SQL	Notes
create, Create, or CREATE, and so on. (any case variation) partition, Partition, PARTITION, and so on (for an Oracle Reserved word)	"CREATE" "PARTITION"	CREATE PARTITION	select "CREATE" select "PARTITION"	Double quotes are required around Oracle SQL Reserved words. These are forced to uppercase, but must always be double-quoted when used anywhere in SQL
rowid, Rowid, ROWid, and so on (for ROWID see notes)	rowid		select "rowid" select "Rowid" select "ROWid" select "rowid"	For ROWID, any mixed or lower-case variation of ROWID preserves the case and must always be double-quoted and use the original case variations. Due to the inherent conflict with Oracle ROWID for the table, if you specify upper-case ROWID, it is automatically stored as lower-case "rowid" and must always be double-quoted when referenced.

① Notes

- In general a column name in an external table can be referenced without double quotes.
- Unless there is an embedded blank, a leading underscore ("_") or leading numeric digit ("0" through "9") in the column name, the original case of the column name is preserved, and it must always be referenced with double quotes and using the original case (upper, lower or mixed-case) of the Avro, ORC, or Parquet column name.
- After using DBMS_CLOUD.CREATE_EXTERNAL_TABLE to create an external table with the format specified as avro, orc, or parquet, use the DESCRIBE command in SQL*Plus to view the table's column names.
- When Oracle SQL Reserved Words are used in Avro, ORC, or Parquet column names, they must always be double-quoted when referenced anywhere in SQL. See *Oracle SQL Reserved Words* in Oracle Database 26ai SQL Language Reference or [Oracle Database 26ai SQL Language Reference](#) for more information.

DBMS_CLOUD Exceptions

The following table describes exceptions for DBMS_CLOUD.

Exception	Code	Description
reject_limit	20003	The reject limit of an external table was reached.
credential_not_exist	20004	A credential object does not exist.
table_not_exist	20005	A table does not exist.
unsupported_obj_store	20006	An unsupported object store URI was provided.
iden_too_long	20008	An identifier is too long.
invalid_format	20009	A format argument is not valid.
missing_credential	20010	Mandatory credential object information was not specified.
invalid_object_uri	20011	An invalid object URI was provided.
invalid_partitioning_clause	20012	An partitioning clause is missing or was not provided.
unsupported_feature	20013	An unsupported feature was used that is not existent in the current database version.
part_not_exist	20014	A partition or subpartition does not exist, or a table is not a partitioned external table or hybrid partitioned table.
invalid_table_name	20016	An invalid table name was used.
invalid_schema_name	20017	An invalid schema name was used.
invalid_dir_name	20018	An invalid directory name was used.
invalid_file_name	20019	An invalid file name was used.
invalid_cred_attribute	20020	Invalid credential attributes were specified.
table_exist	20021	A table already exists.
credential_exist	20022	A credential object already exists.
invalid_req_method	20023	A request method is either too long or invalid.
invalid_req_header	20024	An invalid request header was specified.
file_not_exist	20025	A file does not exist.
invalid_response	20026	An HTTP response was not valid.
invalid_operation	20027	An invalid task class or ID was specified.
invalid_user_name	20028	An invalid username was specified.

Cloud Object Storage URI Formats

This article describes the format of the source file URIs in operations with DBMS_CLOUD. The format depends on the object storage service you are using. DBMS_CLOUD guarantees secure communication and any URI that you specify must use HTTPS, with `https://` as the prefix for the URI.

Oracle Cloud Infrastructure Object Storage Native URI Format

Note

Autonomous AI Database supports only standard-tier storage buckets. It does not support archive-tier storage buckets.

If your source files reside on Oracle Cloud Infrastructure Object Storage in the commercial realm (OC1), it is recommended that you use the following URI format which uses Object Storage Dedicated Endpoints. See [Object Storage Dedicated Endpoints](#), for further information.

```
https://namespace-string.objectstorage.region.oci.customer-oci.com/n/  
namespace-string/b/bucketname/o/filename
```

Note

OCI Object Store dedicated endpoint URLs are only supported in commercial realms (OC1) and available in 19.24 and above.

If your source files reside on Oracle Cloud Infrastructure Object Storage and are not in the commercial realm (OC1), you must use the following format:

```
https://objectstorage.region.oraclecloud.com/n/namespace-string/b/bucket/o/  
filename
```

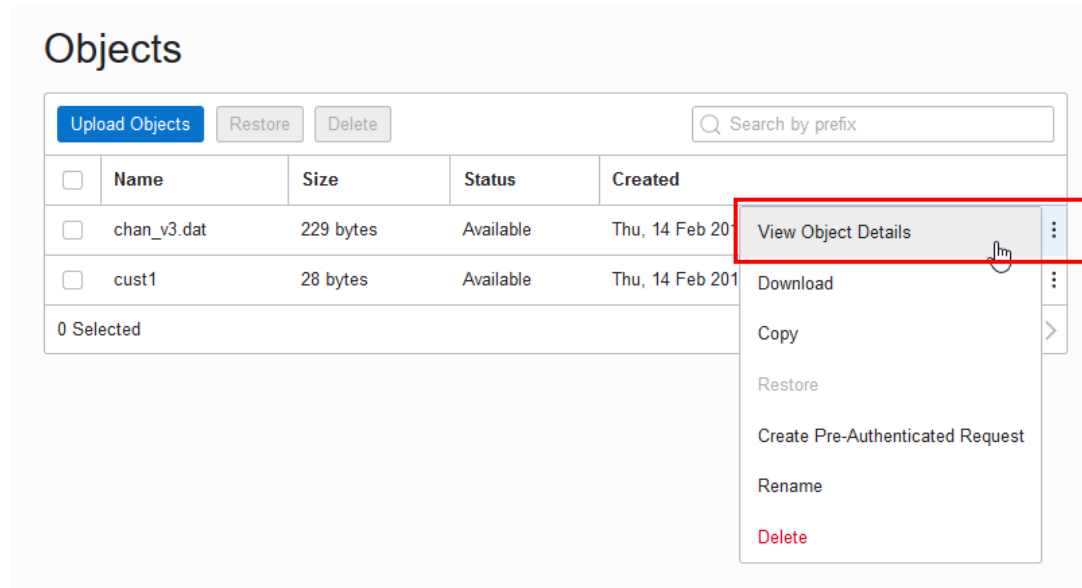
For example, in the commercial realm (OC1) the Native URI for the file `channels.txt` in the `bucketname` bucket in the Phoenix data center is:

```
https://namespace.objectstorage.region.oci.customer-oci.com/n/namespace/b/  
bucketname/o/channels.txt
```

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

You can find the URI from the Oracle Cloud Infrastructure Object Storage "Object Details" in the right hand side ellipsis menu in the Object Store:

1. Open the Oracle Cloud Infrastructure Console by clicking the  next to Oracle Cloud.
2. From the Oracle Cloud Infrastructure left navigation menu click **Core Infrastructure**. Under **Object Storage**, click **Object Storage**.
3. Under List Scope, select a **Compartment**.
4. From the **Name** column, select a bucket.
5. In the Objects area, click **View Object Details**.



6. On the **Object Details** page, the **URL Path (URI)** field shows the URI to access the object.

Oracle Cloud Infrastructure Object Storage Swift URI Format

If your source files reside on Oracle Cloud Infrastructure Object Storage in the commercial realm (OC1), it is recommended that you use the following URI format which uses Object Storage Dedicated Endpoints. See [Object Storage Dedicated Endpoints](#), for further information.

```
https://namespace-string.swiftobjectstorage.region.oci.customer-oci.com/v1/namespace-string/bucket/filename
```

Note

OCI Object Store dedicated endpoint URLs are only supported in commercial realms (OC1) and available in 19.24 and above.

If your source files reside on Oracle Cloud Infrastructure Object Storage and are not in the commercial realm (OC1), you must use the following format:

```
https://swiftobjectstorage.region.oraclecloud.com/v1/namespace-string/bucket/filename
```

For example, in the commercial realm (OC1) the Swift URI for the file `channels.txt` in the `bucketname` bucket in the Phoenix data center is:

```
https://namespace-string.swiftobjectstorage.us-phoenix-1.oci.customer-oci.com/v1/namespace-string/bucketname/channels.txt
```

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

Note

The source files need to be stored in an Object Storage tier bucket. Autonomous AI Database does not support buckets in the Archive Storage tier. See [Overview of Object Storage](#) for more information.

Oracle Cloud Infrastructure Object Storage URI Format Using Pre-Authenticated Request URL

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure pre-authenticated URIs. When you create a pre-authenticated request, a unique URL is generated. You can then provide the unique URL to users in your organization, partners, or third parties to access the Object Storage resource target identified in the pre-authenticated request.

Note

Carefully assess the business requirement for and the security ramifications of pre-authenticated access. When you create the pre-authenticated request URL, note the **Expiration** and the **Access Type** to make sure they are appropriate for your use. A pre-authenticated request URL gives anyone who has the URL access to the targets identified in the request for as long as the request is active. In addition to considering the operational needs of pre-authenticated access, it is equally important to manage its distribution.

If your source files reside on Oracle Cloud Infrastructure Object Storage in the commercial realm (OC1), it is recommended that you use the following URI format which uses Object Storage Dedicated Endpoints. See [Object Storage Dedicated Endpoints](#), for further information.

```
https://namespace-string.objectstorage.region.oci.customer-oci.com/p/encrypted_string/n/  
namespace-string/b/bucket/o/filename
```

Note

OCI Object Store dedicated endpoint URLs are only supported in commercial realms (OC1) and available in 19.24 and above.

If your source files reside on Oracle Cloud Infrastructure Object Storage and are not in the commercial realm (OC1), you must use the following format:

```
https://objectstorage.region.oraclecloud.com.com/p/encrypted_string/n/namespace-string/b/  
bucket/o/filename
```

For example, in the commercial realm (OC1) a sample pre-authenticated URI for the file `channels.txt` in the `bucketname` bucket in the Phoenix data center is:

```
https://namespace-string.objectstorage.us-phoenix-1.oci.customer-oci.com/p/2xN-  
uDtWJNsId910UCYGue/n/namespace-string/b/bucketname/o/channels.txt
```

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

You can use a pre-authenticated URL in any DBMS_CLOUD procedure that takes a URL to access files in Oracle Cloud Infrastructure object store, without the need to create a credential. You need to either specify the `credential_name` parameter as NULL or not supply a `credential_name` parameter.

For example:

```
BEGIN
  DBMS_CLOUD.COPY_DATA(
    table_name => 'CHANNELS',
    file_uri_list => 'https://objectstorage.us-phoenix-1.oraclecloud.com/p/
unique-pre-authenticated-string/n/namespaces-string/b/bucketname/o/
channels.txt',
    format => json_object('delimiter' value ',') );
END;
/
```

Note

A list of mixed URLs is valid. If the URL list contains both pre-authenticated URLs and URLs that require authentication, DBMS_CLOUD uses the specified `credential_name` to access the URLs that require authentication and for the pre-authenticated URLs the specified `credential_name` is ignored.

See [Using Pre-Authenticated Requests](#) for more information.

Amazon S3 URI Format

If your source files reside in Amazon S3, see the following for a description of the URI format for accessing your files: [Accessing a bucket](#).

For example the following refers to the file `channels.txt` in the `atpc` bucket in the `us-west-2` region.

`https://s3-us-west-2.amazonaws.com/atpc/channels.txt`

You can use a presigned URL in any DBMS_CLOUD procedure that takes a URL to access files in Amazon S3 object store, without the need to create a credential. To use a presigned URL in any DBMS_CLOUD procedure, either specify the `credential_name` parameter as NULL, or do not supply a `credential_name` parameter.

See [Share an Object with Others](#) for more information.

Note

DBMS_CLOUD supports the standard Amazon S3 endpoint syntax to access your buckets. DBMS_CLOUD does not support Amazon S3 legacy endpoints. See [Legacy Endpoints](#) for more information.

Azure Blob Storage URI Format

If your source files reside in Azure Blob Storage, see the following for a description of the URI format for accessing your files: [Resource URI Syntax](#).

For example the following refers to the file `channels.txt` in the `atpc` container in the storage account `atpc_user`:

```
https://atpc_user.blob.core.windows.net/atpc/channels.txt
```

Note

You can use Shared Access Signatures (SAS) URL in any `DBMS_CLOUD` procedure that takes a URL to access files in Azure Blob Storage, without the need to create a credential. To use a Shared Access Signature (SAS) URL, either specify the `credential_name` parameter as `NULL`, or do not supply a `credential_name` parameter. See [Grant Limited Access to Azure Storage Resources Using Shared Access Signatures \(SAS\)](#) for more information.

Amazon S3 Compatible URI Format

`DBMS_CLOUD` supports object storage service implementations that support Amazon S3 compatible URLs, including the following services:

- Oracle Cloud Infrastructure Object Storage with Amazon S3 compatible URL
- Google Cloud Storage with Amazon S3 compatible URL
- Wasabi Hot Cloud Storage with Amazon S3 compatible URL

Note

To use `DBMS_CLOUD` with an Amazon S3 compatible object store you need to provide valid credentials. See `CREATE_CREDENTIAL` Procedure for more information.

If your source files reside on a service that supports Amazon S3 compatible URIs, use the following URI format to access your files:

- **Oracle Cloud Infrastructure Object Storage S3 Compatible URL**

If your source files reside on Oracle Cloud Infrastructure Object Storage in the commercial realm (OC1), it is recommended that you use the object URL and bucket URL formats listed below for commercial realm (OC1). See [Object Storage Dedicated Endpoints](#), for further information.

Note

OCI Object Store dedicated endpoint URLs are only supported in commercial realms (OC1) and available in 19.24 and above.

Object URL Formats

- Supported only in the commercial realm (OC1):

```
https://mynamespace.compat.objectstorage.region.oci.customer-oci.com/  
bucket_name/object_name
```

- Supported in all zones:

`https://mynamespace.compat.objectstorage.region.oraclecloud.com/
bucket_name/object_name`

Bucket URL Formats:

- Supported only in the commercial realm (OC1):

`https://mynamespace.compat.objectstorage.region.oci.customer-oci.com/
bucket_name`

- Supported in all zones:

`https://mynamespace.compat.objectstorage.region.oraclecloud.com/
bucket_name`

See [Amazon S3 Compatibility](#) and [Object Storage Service API](#) for more information.

- **Google Cloud Storage S3 Compatible URL**

Object URL Format:

`https://bucketname.storage.googleapis.com/object_name`

Bucket URL Format:

`https://bucketname.storage.googleapis.com/`

See [Migrating from Amazon S3 to Cloud Storage](#) and [Request Endpoints](#) for more information.

- **Wasabi S3 Compatible URL**

Object URL Format:

`https://bucketname.s3.region.wasabisys.com/object_name`

Bucket URL Format:

`https://bucketname.s3.region.wasabisys.com/`

See [Wasabi S3 API Reference](#) and [Service URLs for Wasabi's Storage Regions](#) for more information.

Additional Customer-Managed URI Formats

In addition to the pre-configured, recognized URIs with their fully-qualified domain names (FQDNs), DBMS_CLOUD cannot determine the proper authentication scheme for customer-

managed endpoints URIs. In those cases, DBMS_CLOUD relies on the proper URI scheme to identify the authentication scheme for the customer-managed endpoint.

URI Scheme	Authentication Type	Access Method Description	URI Example
basic://	Basic authentication	Username and password stored in database credential object is used to authenticate the HTTP request	basic://api.github.com/users/myaccount
bearer://	Bearer token authentication	Bearer token stored in the password field in database credential object is used to specify the Authorization header for the HTTP request	bearer://api.sendgrid.com/v3/resource
oci://	OCI native	OCI signing key obtained from database credential object stored and used to sign requests using the OCI authentication protocol	oci://objectstorage.us-ashburn-1.oraclecloud.com
public://	No authentication	Public URLs	public://cms.data.gov/
s3://	Amazon Web Services S3-compatible	Access key and secret key obtained from the username/password field of database credential object, and S3-compatible authentication performed for the HTTP request.	s3://bucket.myprivatesite.com/file1.csv

Examples:

Customer-managed endpoint using S3-compatible authentication.

This example shows how for new URIs, customers can add the public or private host name pattern using DBMS_NETWORK_ACL_ADMIN package. The code block, executed by user ADMIN, enables HTTPS access for user SCOTT to endpoints in domain *.myprivatesite.com. It then shows how user SCOTT accesses the newly enabled endpoint. Note that credential MY_CRED for user SCOTT must store the access key and secret key for S3-compatible authentication performed for the HTTP request indicated by the URI prefix.

```
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => '*.myprivatesite.com',
    ace => xs$ace_type(privilege_list => xs$name_list('http'),
                      principal_name => 'SCOTT',
                      principal_type => xs_acl.ptype_db),
    private_target => TRUE  );
END;
/

BEGIN
  DBMS_CLOUD.get_object(
```

```

        credential_name => 'MY_CRED',
        object_uri      => 's3://bucket.myprivatesite.com/file1.csv',
        directory_name  => 'MY_DIR' );
END;
/

```

Customer-managed endpoint with public access

This example shows how to register the SCOTT user to access public REST APIs. The ADMIN user creates a network ACL for the host to provide access to SCOTT user.

```

BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
        host => 'data.cms.gov',
        ace => xs$ace_type(privilege_list => xs$name_list('http'),
            principal_name => 'SCOTT',
            principal_type => xs_acl.ptype_db)
    );
END;
/

SELECT DBMS_CLOUD.get_response_text(
    DBMS_CLOUD.send_request(
        uri      => 'public://data.cms.gov/provider-data/api/1/datastore/
imports/a',
        method   => DBMS_CLOUD.METHOD_GET,
        headers  => JSON_OBJECT('Accept' VALUE 'application/json')
    )
)
FROM DUAL;
/

```

SODA Collection Metadata on Autonomous AI Database

Describes default and customized collection metadata on Autonomous AI Database.

SODA Default Collection Metadata on Autonomous AI Database

Describes the default collection metadata on Autonomous AI Database, that is the metadata for a collection that is added when custom metadata is not supplied.

Each SODA implementation provides a way to create a default collection when you supply a collection name. For example, in SODA for Java you use the `createCollection` method and supply just a collection name parameter:

```
db.admin().createCollection("myCol");
```

This creates a collection with default collection metadata. When you create a default collection on your Autonomous AI Database, the collection metadata includes the following information (regardless of which SODA implementation you use to create the default collection):

```
{
    "keyColumn" :
```



```

    {
      "name" : "ID",
      "sqlType" : "VARCHAR2",
      "maxLength" : 255,
      "assignmentMethod" : "UUID"
    },

    "contentColumn" :
    {
      "name" : "JSON_DOCUMENT",
      "sqlType" : "BLOB",
      "jsonFormat" : "JSON"
    },

    "versionColumn" :
    {
      "name" : "VERSION",
      "method" : "UUID"
    },

    "lastModifiedColumn" :
    {
      "name" : "LAST_MODIFIED"
    },

    "creationTimeColumn" :
    {
      "name" : "CREATED_ON"
    },

    "readOnly" : false
  }

```

SODA Customized Collection Metadata on Autonomous AI Database

Describes SODA collection custom metadata on Autonomous AI Database.

Each SODA implementation provides a way to customize the collection metadata during collection creation. For example, in SODA for Java, you can use the following command:

```

OracleDocument metadata = db.createDocumentFromString("metadata_string");
OracleCollection col = db.admin().createCollection("myCustomColl", metadata);

```

In this example, for *metadata_string* you can use the default metadata as the starting point, and customize the following:

- Change `keyColumn.assignmentMethod` to `CLIENT`: Change the value of the `assignmentMethod` under `keyColumn` in the metadata to `CLIENT` (instead of `UUID`).
Valid values for `keyColumn.assignmentMethod` on Autonomous AI Database:
 - **UUID** (default): Keys are generated by SODA, based on the UUID.
 - **CLIENT**: Keys are assigned by the client application.
- Provide a `mediaTypeColumn` name value: A media type column is needed if the collection is to be heterogeneous, that is, it can store documents other than JavaScript Object Notation (JSON). See [Media Type Column Name](#) for details.

The following example specifies client-assigned keys and a custom media type column. The `mediaTypeColumn` name is specified with the value `YOUR_MEDIA_TYPE_COLUMN_NAME`. Otherwise, the default settings are used.

```
{
  "keyColumn" :
  {
    "name" : "ID",
    "sqlType" : "VARCHAR2",
    "maxLength" : 255,
    "assignmentMethod" : "CLIENT"
  },

  "contentColumn" :
  {
    "name" : "JSON_DOCUMENT",
    "sqlType" : "BLOB"
  },

  "versionColumn" :
  {
    "name" : "VERSION",
    "method" : "UUID"
  },

  "lastModifiedColumn" :
  {
    "name" : "LAST_MODIFIED"
  },

  "creationTimeColumn" :
  {
    "name" : "CREATED_ON"
  },

  "mediaTypeColumn" :
  {
    "name" : "YOUR_MEDIA_TYPE_COLUMN_NAME"
  },

  "readOnly" : false
}
```

DBMS_CLOUD_REPO Package

DBMS_CLOUD_REPO Overview

The `DBMS_CLOUD_REPO` package provides easy access to files in Cloud Code (Git) Repositories, including: GitHub, AWS CodeCommit, and Azure Repos.

This package is a single interface for access to Multicloud Code repositories and allows you to upload SQL files to Git repositories or install SQL scripts directly from Cloud Code Repositories. This package also allows you to use a Cloud Code Repository to manage code versions for SQL scripts and to install or patch application code from Git repositories.

Concepts

- **Git Version Control System:** [Git](#) is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development. Its goals include speed, data integrity, and support for distributed, non-linear workflows.
- **Git Repository:** A Git repository is a virtual storage of your project. It allows you to save versions of your code, which you can access when needed.

Architecture

DBMS_CLOUD_REPO package provides four feature areas:

- **Repository Initialization with Generic Cloud Code Repository Handle**
 - Initialize a GitHub Code Repository
 - Initialize an AWS CodeCommit Code Repository
 - Initialize an Azure Repos Code Repository
- **Repository Management Operations**
 - Create a repository
 - Update a repository
 - List repositories
 - Delete a repository
- **Repository File Management Operations**
 - Upload a file to Code Repository from Oracle Database.
 - Download a file from Code Repository to Oracle Database.
 - Delete files from Code Repository.
 - List files from Code Repository.
- **SQL Install Operations**
 - Export Database object metadata DDL to repository.
 - Install SQL statements from a file in the Code Repository in Oracle Database.
 - Install SQL statements from a buffer.

DBMS_CLOUD_REPO Data Structures

The DBMS_CLOUD_REPO package defines record types and a generic JSON object type repo.

REPO JSON Object

A DBMS_CLOUD_REPO REPO is an opaque JSON object to represent a Cloud Code Repository of a specific cloud provider. A REPO object can be passed to different DBMS_CLOUD_REPO APIs. This opaque object ensures that DBMS_CLOUD_REPO procedures and functions are multicloud compatible; you do not have to change any code when you migrate from one Cloud Code Repository provider to another Cloud Code Repository.

DBMS_CLOUD_REPO Initialization Operations

Lists the subprograms for initialization operations within the DBMS_CLOUD_REPO package.

Subprogram	Description
INIT_AWS_REPO Function	This function initializes an AWS repository handle and returns an opaque type.
INIT_AZURE_REPO Function	This function initializes an Azure repository handle and returns an opaque type.
INIT_GITHUB_REPO Function	This function initializes a GitHub repository handle and returns an opaque type.
INIT_REPO Function	This function initializes a Cloud Code Repository handle and returns an opaque JSON object.

DBMS_CLOUD_REPO Repository Management Operations

Shows the subprograms for repository management operations within the DBMS_CLOUD_REPO package.

Subprogram	Description
CREATE_REPOSITORY Procedure	This procedure creates a Cloud Code Repository identified by the repo handle argument.
DELETE_REPOSITORY Procedure	This procedure deletes the Cloud Code Repository identified by the repo handle argument.
LIST_REPOSITORIES Function	This function lists all the Cloud Code Repositories identified by the repo handle argument.
UPDATE_REPOSITORY Procedure	This procedure updates a Cloud Code repository identified by the repo handle argument. The procedure supports updating the name, description, or private visibility status, as supported by the Cloud Code repository.

DBMS_CLOUD_REPO Repository Branch Management Operations

Lists the subprograms for repository branch management operations within the DBMS_CLOUD_REPO package.

Subprogram	Description
CREATE_BRANCH Procedure	This procedure creates a branch in a Cloud Code Repository identified by the repo handle argument.
DELETE_BRANCH Procedure	This procedure deletes a branch in a Cloud Code Repository identified by the repo handle argument.
LIST_BRANCHES Function	This function lists all the Cloud Code Repository branches identified by the repo handle argument.
LIST_COMMITS Function	This function lists all the commits in a Cloud Code Repository branch identified by the repo handle argument.
MERGE_BRANCH Procedure	This procedure merges a Cloud Code Repository branch into another specified branch in a Cloud Code Repository identified by the repo handle argument.

DBMS_CLOUD_REPO File Operations

Lists the subprograms for file operations within the DBMS_CLOUD_REPO package.

Subprogram	Description
DELETE_FILE Procedure	This procedure deletes a file from the Cloud Code repository identified by the repo handle argument.

Subprogram	Description
GET_FILE Procedure and Function	The function downloads the contents of a file from the Cloud Code repository. The procedure allows you to download the contents of a file from the Cloud Code repository and save the file in a directory.
LIST_FILES Function	This function downloads a file from Cloud Code repository. Optionally, file content can be accessed from either a specific branch, tag or commit name. By default, the file is accessed from the default repository branch.
PUT_FILE Procedure	This procedure uploads a file to the Cloud Code repository identified by the repo handle argument. The procedure is overloaded to support either uploading a file from a directory object or uploading the contents from a CLOB to the repository file.

DBMS_CLOUD_REPO SQL Install Operations

Lists the subprograms for SQL install operations within the DBMS_CLOUD_REPO package.

Subprogram	Description
EXPORT_OBJECT Procedure	This procedure uploads the DDL metadata of a database object to the Cloud Code repository identified by the repo handle argument.
EXPORT_SCHEMA Procedure	This procedure exports metadata of all objects in a schema to a Cloud Code Repository branch identified by the repo handle argument.
INSTALL_FILE Procedure	This procedure installs SQL statements from a file in the Cloud Code repository identified by the repo handle argument.
INSTALL_SQL Procedure	This procedure installs SQL statements from a buffer given as input.

Summary of DBMS_CLOUD_REPO Subprograms

This section covers the DBMS_CLOUD_REPO subprograms provided with Autonomous AI Database.

The DBMS_CLOUD_REPO package is made up of the following:

- DBMS_CLOUD_REPO Initialization Operations
- DBMS_CLOUD_REPO Repository Management Operations
- DBMS_CLOUD_REPO File Operations
- DBMS_CLOUD_REPO SQL Install Operations

Prerequisites

As a developer, you can use DBMS_CLOUD procedures with Autonomous AI Databases deployed on Oracle Public Cloud, Multicloud or Exadata Cloud@Customer.

Depending on the deployment choice, the following prerequisites must be met to use the DBMS_CLOUD procedures with Amazon S3, Azure Blob Storage, and Google Cloud Storage service providers.

-
- [Oracle Public Cloud or Multicloud](#)
 - [Exadata Cloud@Customer](#)

Oracle Public Cloud or Multicloud

An outbound connectivity must have been configured using a NAT gateway, by your fleet administrator as described below:

- Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
- After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 1. Go to the **Subnet Details** page for the subnet.
 2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
 3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** NAT Gateway
 - **Target:** The name of the NAT gateway you just created in the VCNIf such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.
 4. Return to the **Subnet Details** page for the subnet.
 5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
 6. In the side menu, under **Resources**, click **Egress Rules**.
 7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** AllIf such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

CREATE_BRANCH Procedure

This procedure creates a branch in the Cloud Code Repository identified by the repo handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.CREATE_BRANCH(  
    repo            IN    CLOB,  
    branch_name     IN    VARCHAR2,  
    parent_branch_name IN  VARCHAR2 DEFAULT NULL,  
    parent_commit_id IN  VARCHAR2 DEFAULT NULL  
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.
branch_name	Specifies the repository branch name. This parameter is mandatory and supported for all cloud providers.
parent_branch_name	Creates the new branch using the head commit of the specified parent branch. This parameter is supported for all cloud providers. If you do not supply a parent_branch_name value, the parent_branch_name is set to main.
parent_commit_id	Creates the new branch using the specified repository commit. This parameter is supported for all cloud providers. If you do not supply a parent_commit_id value, the parent_commit_id is set to a NULL value.

Note

To create a branch in a Cloud Code repository, you must specify the parent branch or the parent commit id.

Example

```
BEGIN
  DBMS_CLOUD_REPO.CREATE_BRANCH (
    repo           => l_repo,
    branch_name    => 'test_branch',
    parent_branch_name => 'main'
  );
END;
/
```

Usage Note

To run the `DBMS_CLOUD_REPO.CREATE_BRANCH` procedure, you must be logged in as the `ADMIN` user or have `EXECUTE` privilege on `DBMS_CLOUD_REPO`.

CREATE_REPOSITORY Procedure

This procedure creates a Cloud Code Repository identified by the `repo` handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.CREATE_REPOSITORY(
    repo           IN    CLOB,
    description    IN    CLOB    DEFAULT NULL,
    private        IN    BOOLEAN DEFAULT TRUE
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is supported for all cloud providers.
description	A short text description for the repository. This parameter is supported for GITHUB and AWS cloud provider.
private	Repository is private and only accessible with valid credentials This parameter is only supported for the GITHUB cloud provider.

Example

```
BEGIN
  DBMS_CLOUD_REPO.CREATE_REPOSITORY(
    repo           => l_repo,
    description    => 'My test repo',
    private        => TRUE
  );
END;
/
```


DELETE_BRANCH Procedure

This procedure deletes a branch in the Cloud Code repository identified by the repo handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.DELETE_BRANCH (  
    repo          IN CLOB,  
    branch_name   IN VARCHAR2 DEFAULT NULL  
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.
branch_name	Delete branch from a specific repository. This parameter is mandatory and supported for all cloud providers.

Example

```
BEGIN  
    DBMS_CLOUD_REPO.DELETE_BRANCH (  
        repo      => l_repo,  
        branch_name => 'test_branch'  
    );  
END;  
/
```

Usage Note

To run the DBMS_CLOUD_REPO.DELETE_BRANCH procedure, you must be logged in as the ADMIN user or have EXECUTE privilege on DBMS_CLOUD_REPO.

DELETE_FILE Procedure

This procedure deletes a file from the Cloud Code repository identified by the repo handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.DELETE_FILE(  
    repo          IN CLOB,  
    file_path     IN VARCHAR2,  
    branch_name   IN VARCHAR2 DEFAULT NULL,  
    commit_details IN CLOB      DEFAULT NULL  
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path to delete file in the repository.
branch_name	Delete file from a specific branch.
commit_details	Commit Details as a JSON document { "message": "Commit message", "author": { "name": "Committing user name", "email": "Email of committing user" } }

Example

```
BEGIN
  DBMS_CLOUD_REPO.DELETE_FILE(
    repo      => l_repo,
    file_path  => 'scripts/test3.sql',
    branch_name => 'test_branch'
  );
END;
/
```

DELETE_REPOSITORY Procedure

This procedure deletes the Cloud Code Repository identified by the repo handle argument.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.DELETE_REPOSITORY(
    repo          IN    CLOB
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.

Example

```
BEGIN
  DBMS_CLOUD_REPO.DELETE_REPOSITORY(
    repo => l_repo
  );
END;
/
```

EXPORT_OBJECT Procedure

This procedure uploads the DDL metadata of a database object to the Cloud Code repository identified by the repo handle argument. This procedure is an easy way to upload the metadata definition of a database object in single step.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.EXPORT_OBJECT(
    repo          IN CLOB,
    file_path     IN VARCHAR2,
    object_type   IN VARCHAR2,
    object_name   IN VARCHAR2 DEFAULT NULL,
    object_schema IN VARCHAR2 DEFAULT NULL,
    branch_name   IN VARCHAR2 DEFAULT NULL,
    commit_details IN CLOB      DEFAULT NULL,
    append        IN BOOLEAN   DEFAULT FALSE
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path to upload object metadata in the repository.
object_type	Object type supported by DBMS_METADATA. See <i>DBMS_METADATA: Object Types</i> table in Oracle Database 19c PL/SQL Packages and Types Reference or Oracle Database 26ai PL/SQL Packages and Types Reference for details.
object_name	Name of the database object to retrieve metadata.
object_schema	Owning schema of the database object.
branch_name	Put file to a specific branch.
commit_details	Commit Details as a JSON document:{"message": "Commit message", "author": {"name": "Committing user name", "email": "Email of committing user" } }
append	Append metadata DDL to existing file.

Usage Note

For customized control on the object DDL, you can use DBMS_METADATA.GET_DDL along with DBMS_CLOUD_REPO.PUT_FILE. In order to get metadata definition of the object, the current user must be privileged to retrieve the object metadata. See *DBMS_METADATA* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for the security requirements of the package.

Example

```
BEGIN
    DBMS_CLOUD_REPO.EXPORT_OBJECT(
        repo          => l_repo,
        object_type   => 'PACKAGE',
        object_name   => 'MYPACK',
```

```

        file_path => 'mypack.sql'
    );
END;
/

```

EXPORT_SCHEMA Procedure

This procedure exports metadata of all objects in a schema to the Cloud Code Repository branch identified by the repo handle argument.

Syntax

```

PROCEDURE DBMS_CLOUD_REPO.EXPORT_SCHEMA(
    repo           IN CLOB,
    file_path      IN VARCHAR2,
    schema_name    IN VARCHAR2,
    filter_list     IN CLOB          DEFAULT NULL,
    branch_name    IN VARCHAR2     DEFAULT NULL,
    commit_details IN CLOB          DEFAULT NULL
);

```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.
file_path	Specifies the name of the schema file to upload to the repo. This parameter is mandatory and supported for all cloud providers.
schema_name	Specifies the name of the schema for which a DDL script is to be uploaded to the Cloud Code Repository branch. This parameter is mandatory and supported for all cloud providers.
filter_list	Specifies the CLOB of the JSON array that defines the filter conditions to include or exclude the objects whose metadata needs to be exported. This parameter is supported for all cloud providers. JSON parameters for filter_list are: - match_type: Specifies the type of filter to be applied to the object types or object names. Valid match_type values are: - in/not_in - like/not_like - equal/not_equal - type: Specifies the type of object by which to filter. - name: Specifies the name of the object by which to filter.
branch_name	Specifies the repository branch name. This parameter is supported for all cloud providers. If you do not supply a branch_name value, the branch_name is set to the default repository branch.

Parameter	Description
commit_details	Commit Details as a JSON document <pre>{ "message": "Commit message", "author": { "name": "Committing user name", "email": "Email of committing user" } }</pre> This parameter is supported for all cloud providers. If you do not supply a commit_details value, the commit_details is set to the default commit message that includes the information about the current database session user and database name performing the commit.

Example

```
BEGIN
  DBMS_CLOUD_REPO.EXPORT_SCHEMA(
    repo          => l_repo,
    schema_name    => 'USER1',
    file_path      => 'myschema_ddl.sql'
    filter_list    =>
      to_clob('[
        { "match_type": "equal",
          "type": "table"
        },
        { "match_type": "not_equal",
          "type": "view"
        },
        { "match_type": "in",
          "type": "table",
          "name": " 'EMPLOYEE_SALARY' ', 'EMPLOYEE_ADDRESS' ' "
        },
        { "match_type": "equal",
          "type": "sequence",
          "name": "EMPLOYEE_RECORD_SEQ"
        },
        { "match_type": "like",
          "type": "table",
          "name": "%OFFICE%"
        }
      ]'
    );
END;
```

Usage Note

To run the `DBMS_CLOUD_REPO.EXPORT_SCHEMA` procedure, you must be logged in as the `ADMIN` user or have `EXECUTE` privilege on `DBMS_CLOUD_REPO`.

GET_FILE Procedure and Function

The function downloads the contents of a file from the Cloud Code repository. The procedure allows you to download the contents of a file from the Cloud Code repository and save the file in a directory.

Syntax

```

FUNCTION DBMS_CLOUD_REPO.GET_FILE(
    repo          IN  CLOB,
    file_path      IN  VARCHAR2,
    branch_name    IN  VARCHAR2  DEFAULT NULL,
    tag_name       IN  VARCHAR2  DEFAULT NULL,
    commit_name    IN  VARCHAR2  DEFAULT NULL
) RETURN CLOB;

PROCEDURE DBMS_CLOUD_REPO.GET_FILE(
    repo          IN  CLOB,
    file_path      IN  VARCHAR2,
    directory_name IN  VARCHAR2,
    target_file_name IN VARCHAR2  DEFAULT NULL,
    branch_name    IN  VARCHAR2  DEFAULT NULL,
    tag_name       IN  VARCHAR2  DEFAULT NULL,
    commit_name    IN  VARCHAR2  DEFAULT NULL
);

```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path in the repository.
directory_name	Directory object name to save the file contents.
target_file_name	Target file name to save contents in directory.
branch_name	Get file from a specific branch.
tag_name	Get file from a specific Tag.
commit_name	Get file from a specific commit.

Example

```

BEGIN
    DBMS_CLOUD_REPO.GET_FILE(
        repo          => l_repo,
        file_path      => 'test3.sql',
        directory_name => 'DATA_PUMP_DIR',
        target_file_name => 'test2.sql'
    );
END;
/

```

INIT_AWS_REPO Function

This function initializes an AWS repository handle and returns an opaque type.

Syntax

```

FUNCTION DBMS_CLOUD_REPO.INIT_AWS_REPO(
    credential_name IN  VARCHAR2,

```

```

        repo_name      IN  VARCHAR2,
        region         IN  VARCHAR2
    ) RETURN repo;

```

Parameters

Parameter	Description
credential_name	Credential object specifying AWS CodeCommit accesskey/secretkey.
repo_name	Specifies the repository name.
region	Specifies the AWS region for the CodeCommit repository.

Example

```

BEGIN
    :repo := DBMS_CLOUD_REPO.INIT_AWS_REPO(
        credential_name => 'AWS_CRED',
        repo_name       => 'my_repo',
        region          => 'us-east-1'
    );
END;
/

```

INIT_AZURE_REPO Function

This function initializes an Azure repository handle and returns an opaque type. This function is only supported for Azure cloud provider.

Syntax

```

FUNCTION DBMS_CLOUD_REPO.INIT_AZURE_REPO(
    credential_name IN  VARCHAR2,
    repo_name      IN  VARCHAR2,
    organization   IN  VARCHAR2,
    project        IN  VARCHAR2
) RETURN repo;

```

Parameters

Parameter	Description
credential_name	Credential object specifying Azure, with a Username and Personal Access Token (PAT).
repo_name	Specifies the repository name.
organization	Specifies the Azure DevOps Organization.
project	Azure Team Project name.

Example

```

BEGIN
    :repo := DBMS_CLOUD_REPO.INIT_AZURE_REPO(
        credential_name => 'AZURE_CRED',
        repo_name       => 'my_repo',

```

```
        organization => 'myorg',  
        project      => 'myproj',  
    );  
END;  
/
```

INIT_GITHUB_REPO Function

This function initializes a GitHub repository handle and returns an opaque type.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.INIT_GITHUB_REPO(  
    credential_name IN VARCHAR2 DEFAULT NULL,  
    repo_name      IN VARCHAR2,  
    owner          IN VARCHAR2)  
RETURN repo;
```

Parameters

Parameter	Description
credential_name	Credential object specifying GitHub. User Email and Personal Access Token (PAT).
repo_name	Specifies the repository name.
owner	Specifies the repository owner.

Example

```
BEGIN  
    :repo := DBMS_CLOUD_REPO.INIT_GITHUB_REPO(  
        credential_name => 'GITHUB_CRED',  
        repo_name      => 'my_repo',  
        owner          => 'foo'  
    );  
END;  
/
```

INIT_REPO Function

This function initializes a Cloud Code Repository handle and returns an opaque JSON object. This function is a generic interface to accept a JSON document, and avoids having to change code, you only need to change a JSON document, when moving a code repository from one Cloud Code repository to another Cloud Code repository.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.INIT_REPO(  
    params IN CLOB)  
RETURN CLOB;
```


Parameters

JSON Parameter	Description
provider	Cloud code repository provider from the following: DBMS_CLOUD_REPO.GITHUB_REPO ('GITHUB') DBMS_CLOUD_REPO.AWS_REPO ('AWS') DBMS_CLOUD_REPO.AZURE_REPO ('AZURE')
repo_name	Specifies the repository name. DBMS_CLOUD_REPO.PARAM_REPO_NAME
owner	GitHub Repository Owner. DBMS_CLOUD_REPO.PARAM_OWNER This parameter is only applicable for GitHub cloud provider.
region	AWS Repository Region DBMS_CLOUD_REPO.PARAM_REGION This parameter is only applicable for AWS cloud provider.
organization	Azure Organization DBMS_CLOUD_REPO.PARAM_ORGANIZATION This parameter is only applicable for Azure cloud provider.
project	Azure Team Project DBMS_CLOUD_REPO.PARAM_PROJECT This parameter is only applicable for Azure cloud provider

Example

```

BEGIN
  :repo := DBMS_CLOUD_REPO.INIT_REPO(
    params => JSON_OBJECT('credential_name' value 'mycred',
                          'repo_name'       value 'myrepo',
                          'repo_owner'      value 'foo')
  );
END;
/

```

INSTALL_FILE Procedure

This procedure installs SQL statements from a file in the Cloud Code repository identified by the repo handle argument.

Syntax

```

PROCEDURE DBMS_CLOUD_REPO.INSTALL_FILE(
  repo           IN CLOB,
  file_path      IN VARCHAR2,
  branch_name    IN VARCHAR2 DEFAULT NULL,
  tag_name       IN VARCHAR2 DEFAULT NULL,
  commit_name    IN VARCHAR2 DEFAULT NULL,
  stop_on_error  IN BOOLEAN  DEFAULT TRUE
);

```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path in the repository.

Parameter	Description
branch_name	Branch to install file from a specific branch.
tag_name	Tag to install file from a specific Tag.
commit_name	Commit ID to install file from a specific commit.
stop_on_error	Stop executing the SQL statements on first error.

Usage Notes

- You can install SQL statements containing nested SQL from a Cloud Code repository file using the following:
 - @: includes a SQL file with a relative path to the ROOT of the repository.
 - @@: includes a SQL file with a path relative to the current file.
- The scripts are intended as schema install scripts and not as generic SQL scripts:
 - Scripts cannot contain SQL*Plus client specific commands.
 - Scripts cannot contain bind variables or parameterized scripts.
 - SQL statements must be terminated with a slash on a new line (/).
 - Scripts can contain DDL, DML PLSQL statements, but direct SELECT statements are not supported. Using SELECT within a PL/SQL block is supported.

Any SQL statement that can be run using EXECUTE IMMEDIATE will work if it does not contain bind variables or defines.

Example

```
BEGIN
    DBMS_CLOUD_REPO.INSTALL_FILE(
        repo          => l_repo,
        file_path      => 'test3.sql',
        stop_on_error => FALSE
    );
END;
/
```

INSTALL_SQL Procedure

This procedure installs SQL statements from a buffer given as input.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.INSTALL_SQL(
    content          IN CLOB,
    stop_on_error    IN BOOLEAN DEFAULT TRUE
);
```

Parameters

Parameter	Descriptions
content	Is the CLOB containing the SQL statements to run.

Parameter	Descriptions
stop_on_error	Stop executing the SQL statements on first error.

Usage Notes

- The scripts are intended as schema install scripts and not as generic SQL scripts:
 - Scripts cannot contain SQL*Plus client specific commands.
 - Scripts cannot contain bind variables or parameterized scripts.
 - SQL statements must be terminated with a slash on a new line (/).
 - Scripts can contain DDL, DML PLSQL statements, but direct SELECT statements are not supported. Using SELECT within a PL/SQL block is supported.

Any SQL statement that can be run using EXECUTE IMMEDIATE will work if it does not contain bind variables or defines.

Example

```
BEGIN
  DBMS_CLOUD_REPO.INSTALL_SQL(
    content => 'create table t1 (x varchar2(30))' || CHR(10) || '/',
    stop_on_error => FALSE
  );
END;
/
```

LIST_BRANCHES Function

This function lists branches in the Cloud Code Repository branch identified by the repo handle argument.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.LIST_BRANCHES(
  repo          IN    CLOB
) RETURN list_branch_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.

Example

```
SELECT * FROM DBMS_CLOUD_REPO.LIST_BRANCHES (repo => l_repo);
```

Usage Notes

- This is a pipelined table function with return type as list_branch_ret_tab.

- `DBMS_CLOUD_REPO.LIST_BRANCHES` returns the column: `name`, which indicates the name of the Cloud Code Repository branch.

LIST_COMMITS Function

This function lists commits in the Cloud Code Repository branch identified by the `repo` handle argument.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.LIST_COMMITS(
    repo          IN CLOB,
    branch_name   IN VARCHAR2 DEFAULT NULL,
    file_path     IN VARCHAR2 DEFAULT NULL,
    commit_id     IN VARCHAR2 DEFAULT NULL
) RETURN list_commit_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

Parameter	Description
<code>repo</code>	Specifies the repository handle. This parameter is mandatory and supported for all cloud providers.
<code>branch_name</code>	List commits from a specific branch. This parameter is supported for all cloud providers. If you do not supply a <code>branch_name</code> value, the <code>branch_name</code> is set to <code>main</code> .
<code>file_path</code>	List files under the specified subfolder path in the repository. This parameter is only supported for Git and Azure cloud providers. If you do not supply a <code>file_path</code> value, the <code>file_path</code> is set to a <code>NULL</code> value.
<code>commit_id</code>	List files starting from the specified <code>sha/id</code> This parameter is supported for all cloud providers. If you do not supply a <code>commit_id</code> value, the <code>commit_id</code> is set to a <code>NULL</code> value.

Example

```
SELECT name FROM DBMS_CLOUD_REPO.LIST_COMMITS(repo => l_repo);
```

Example

```
SELECT name FROM DBMS_CLOUD_REPO.LIST_COMMITS (
    repo          => l_repo,
    commit_id     => '66dd2b23b74cd0afabd11af66c6aa9c550540ba6',
    file_path     => 'sub_dir/test11.sql'
);
```

Usage Notes

- This is a pipelined table function with a return type as `list_commit_ret_tab`.
- `DBMS_CLOUD_REPO.LIST_COMMITS` returns the column: `commit_id`.

LIST_FILES Function

This function downloads a file from Cloud Code repository. Optionally, file content can be accessed from either a specific branch, tag or commit name. By default, the file is accessed from the default repository branch. The results include the file names and additional metadata about the files.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.LIST_FILES(  
    repo          IN CLOB,  
    path          IN VARCHAR2 DEFAULT NULL,  
    branch_name   IN VARCHAR2 DEFAULT NULL,  
    tag_name      IN VARCHAR2 DEFAULT NULL,  
    commit_id     IN VARCHAR2 DEFAULT NULL  
) RETURN list_file_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

Parameter	Description
repo	Specifies the repository handle.
path	List files under the specified subfolder path in the repository.
branch_name	List files from a specific branch.
tag_name	List files from a specific Tag.
commit_name	List files from a specific commit.

Usage Notes

- This is a pipelined table function with return type as `list_file_ret_tab`.
- `DBMS_CLOUD_REPO.LIST_FILES` returns the columns: `id`, `name`, `url`, and `bytes`.

Example

```
SELECT name FROM DBMS_CLOUD_REPO.LIST_FILES(repo => l_repo);
```

```
NAME  
-----  
test3.sql
```

LIST_REPOSITORIES Function

This function lists all the Cloud Code Repositories identified by the `repo` handle argument. The results include the repository names and additional metadata about the repositories.

Syntax

```
FUNCTION DBMS_CLOUD_REPO.LIST_REPOSITORIES(  
    repo          IN CLOB  
) RETURN list_repo_ret_tab PIPELINED PARALLEL_ENABLE;
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is supported by all cloud providers.
description	A short text description for the repository. This parameter is supported the GITHUB and AWS cloud providers.
private	Repository is private and only accessible with valid credentials This parameter is supported for the GITHUB cloud provider.

Usage Notes

- This is a pipelined table function with return type as list_repo_ret_tab.
- DBMS_CLOUD_REPO.LIST_REPOSITORIES returns the columns: id, name, owner, description, private, url, bytes, created, and last_modified.

Example

```
SELECT name description FROM DBMS_CLOUD_REPO.LIST_REPOSITORIES(:repo);
```

NAME	DESCRIPTION
-----	-----
TestRepo1	My test repo

MERGE_BRANCH Procedure

This procedure merges a repository branch into another specified branch in the Cloud Code Repository identified by the repo handle argument. The MERGE_BRANCH procedure is currently not supported in Azure.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.MERGE_BRANCH (
    repo           IN    CLOB,
    branch_name    IN    VARCHAR2,
    parent_branch_name IN  VARCHAR2 DEFAULT NULL,
    commit_details IN    CLOB      DEFAULT NULL
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is mandatory and supported for GITHUB and AWS cloud providers.
branch_name	Specifies the Git branch name to merge. This parameter is mandatory and supported for all cloud providers.
target_branch_name	Specifies the target branch name to merge into. This parameter is mandatory and supported for all cloud providers.

Parameter	Description
commit_details	Commit Details as a JSON document <pre>{ "message": "Commit message", "author": { "name": "Committing user name", "email": "Email of committing user" } }</pre> If you do not supply a commit_detailsvalue, the commit_details is set to the default commit message that includes the information about current database session user and database name performing the commit.

Example

```
BEGIN
  DBMS_CLOUD_REPO.MERGE_BRANCH (
    repo          => l_repo,
    branch_name    => 'test_branch',
    target_branch_name => 'main'
  );
END;
/
```

Usage Note

To run the DBMS_CLOUD_REPO.MERGE_BRANCH procedure, you must be logged in as the ADMIN user or have EXECUTE privilege on DBMS_CLOUD_REPO.

PUT_FILE Procedure

This procedure uploads a file to the Cloud Code repository identified by the repo handle argument. The procedure is overloaded to support either uploading a file from a directory object or uploading the contents from a BLOB to the repository file.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.PUT_FILE(
  repo          IN CLOB,
  file_path     IN VARCHAR2,
  contents      IN BLOB,
  branch_name   IN VARCHAR2 DEFAULT NULL,
  commit_details IN CLOB     DEFAULT NULL
);
```

```
PROCEDURE DBMS_CLOUD_REPO.PUT_FILE(
  repo          IN CLOB,
  file_path     IN VARCHAR2,
  directory_name IN VARCHAR2,
  source_file_name IN VARCHAR2 DEFAULT NULL,
  branch_name   IN VARCHAR2 DEFAULT NULL,
  commit_details IN CLOB     DEFAULT NULL
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle.
file_path	File path to upload file in the repository.
contents	BLOB containing the file contents.
directory_name	Directory object name containing the file name.
source_file_name	Source file name to upload to repository.
branch_name	Put file to a specific branch.
commit_details	Commit Details as a JSON document: { "message": "Commit message", "author": { "name": "Committing user name", "email": "Email of committing user" } }

Example

```
BEGIN
  DBMS_CLOUD_REPO.PUT_FILE(
    repo => l_repo,
  );
END;
/
```

UPDATE_REPOSITORY Procedure

This procedure updates a Cloud Code repository identified by the repo handle argument. UPDATE_REPOSITORY supports updating the name, description, or private visibility status, as supported by the Cloud Code repository.

Syntax

```
PROCEDURE DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
  repo          IN OUT CLOB,
  new_name      IN      VARCHAR2 DEFAULT NULL,
  description    IN      CLOB      DEFAULT NULL,
  private       IN      BOOLEAN   DEFAULT NULL
);
```

Parameters

Parameter	Description
repo	Specifies the repository handle. This parameter is supported for all cloud providers.
new_name	New name for repository. This parameter is supported for all cloud providers.
description	A short text description for the repository. This parameter is supported for GITHUB and AWS cloud providers.
private	Repository is private and only accessible with valid credentials. This parameter is supported for the GITHUB cloud provider.

Example

```
BEGIN
  DBMS_CLOUD_REPO.UPDATE_REPOSITORY(
    repo      => l_repo,
    new_name   => 'repo2'
  );
END;
/
```

DBMS_DCAT Package

About DBMS_DCAT Package

The DBMS_DCAT package provides functions and procedures to help Autonomous AI Database users leverage the data discovery and centralized metadata management system of OCI Data Catalog.

Note

Support for DBMS_DCAT is available in Oracle AI Database 19c starting with version 19.30, and in Autonomous AI Database 26ai starting with version 23.26.1.

Data Catalog harvests metadata from a data lake's object storage assets. The harvesting process creates logical entities, which can be thought of as tables with columns and associated data types. DBMS_DCAT procedures and functions connect Autonomous AI Database to Data Catalog and then synchronize the assets with the database, creating protected schemas and external tables. You can then query object store using those external tables, easily joining external data with data stored in Autonomous AI Database. This dramatically simplifies the management process; there is a single, centrally managed metadata store that is shared across multiple OCI services (including Autonomous AI Databases). There are also Autonomous AI Database dictionary views that allow you to inspect the contents of Data Catalog using SQL, and show you how these Data Catalog entities map to your Autonomous AI Database schemas and tables.

Data Catalog Users and Roles

The DBMS_DCAT package supports synced users/schemas, dcat_admin users and local users. Users must have the dcat_sync role to be able to use this package.

Data Catalog Users

- Synced users/schemas
The synced external tables are organized into database schemas corresponding to Data Asset/Bucket combinations, or according to custom properties set by the user. The synced schemas are automatically created/dropped during Data Catalog synchronization. They are created as no authentication users without the CREATE SESSION privilege. The synced schemas are also created using the protected clause, so that they cannot be altered by local users (not even the PDB admin) and can only be modified through the synchronization.
- User dcat_admin

User `dcat_admin` is a local database user that can run a sync and grant READ privilege on synced tables to other users or roles. The user is created as a no authentication user without the CREATE SESSION privilege.

- **Local users**
Database users querying the external tables must be explicitly granted READ privileges on the synced external tables by users `dcat_admin` or `ADMIN`. By default, after the sync is completed, only users `dcat_admin` and `ADMIN` have access to the synced external tables.

Data Catalog Roles

- **dcat_sync**
The `dcat_sync` role has all the required privileges for using the DBMS_DCAT package. Users must have this role to be able to use the API for navigating the Data Catalog and running the sync.

Required Credentials and IAM Policies

This topic describes the Oracle Cloud Infrastructure Identity and Access Management (IAM) user credentials and policies required to give Autonomous AI Database users permission to manage a data catalog and to read from object storage.

OCI Data Catalog Credential and Policy Requirements:

- A credential object with permission to manage a Data Catalog instance is required. Credential objects containing OCI native authentication is supported. Credential objects based on authentication token user principals and resource principals credentials are not supported.
For information on managing credentials, see DBMS_CLOUD for Access Management.

For OCI native authentication examples, see [Example: Creating an OCI Native Authentication Credential Object](#) and [Autonomous AI Database Now Supports Accessing the Object Storage with OCI Native Authentication](#).

- The manage Data Catalog privilege is required in order for Autonomous AI Database to add custom properties to the Data Catalog namespace. These privileges allow you to override schema names, table names, column names and more.

For further information on Data Catalog permissions, see [Permissions Required for Each API Operation](#).

- The read object storage privilege on buckets is required so that Autonomous AI Database can query data files.

For further Oracle Object Storage Policy Examples, see [Policy Examples](#).

AWS Glue Data Catalog Credential and Policy Requirements

The following user credentials and policies are required to give Autonomous AI Database users permission to access Amazon Web Services (AWS) Glue Data Catalogs and to read from the S3 object storage:

- A credential object with permission to access an AWS Glue Data Catalog is required. For information on managing credentials, see DBMS_CLOUD for Access Management. For accessing an AWS Glue Data Catalog the following privileges are required: `glue:GetDatabases`, `glue:GetTables`, and `glue:GetTable`.

In addition, privilege `s3:GetBucketLocation` is needed during synchronization for generating resolvable https urls pointing to the underlying S3 objects.

- A credential object with permission to access the files stored in S3 is required so that Autonomous AI Database can query data files.
- AWS credentials are supported. AWS Amazon Resource Names (ARN) credentials are not supported.

Example: Creating an OCI Native Authentication Credential Object

In this example, we create an OCI native authentication credential that can be used when creating a data catalog or an object store credential object. For more details, see `DBMS_DCAT.SET_DATA_CATALOG_CREDENTIAL` Procedure and `DBMS_DCAT.SET_OBJECT_STORE_CREDENTIAL` Procedure respectively.

In OCI native authentication, the `DBMS_CLOUD.CREATE_CREDENTIAL` procedure includes these parameters: `credential_name`, `user_ocid`, `tenancy_ocid`, `private_key`, and `fingerprint`. See `DBMS_CLOUD.CREATE_CREDENTIAL` Procedure for a complete description of this procedure.

The `credential_name` is the name of the credential object. The `user_ocid` and `tenancy_ocid` parameters correspond to the user's and tenancy's OCIDs respectively.

The `private_key` parameter specifies the generated private key in PEM format. Private keys created with a passphrase are not supported. Therefore, we need to make sure we generate a key with no passphrase. See [How to Generate an API Signing Key](#) for more details on how to create a private key with no passphrase. Also, the private key that we provide for this parameter must only contain the key itself without any header or footer (e.g. '-----BEGIN RSA PRIVATE KEY-----', '-----END RSA PRIVATE KEY-----').

The `fingerprint` parameter specifies the fingerprint that is obtained either after uploading the public key to the console, or using the OpenSSL commands. See [How to Upload the Public Key](#) and [How to Get the Key's Fingerprint](#) for further details on obtaining the fingerprint.

Once all the necessary information is gathered and the private key is generated, we're ready to run the following `CREATE_CREDENTIAL` procedure:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL (
    credential_name => 'OCI_NATIVE_CRED',
    user_ocid      =>
'ocid1.user.oc1..aaaaaaaatfn77fe3fxux3o5lego7glqjejrzsqsrs64f4jsjrhbsk5qzndq'
  ,
    tenancy_ocid   =>
'ocid1.tenancy.oc1..aaaaaaaapwkfqz3upqklvmelbm3j77nn3y7uqmlsod75rea5zmtmb1574v
e6a',
    private_key    =>
'MIIEogIBAAKCAQEA...t9SH7Zx7a5iV7QZJS5WeFLMUEv+YbYAjnXK+d0nPQtKh0blQWCEY3Hsblj
7Xz7o=',
    fingerprint    =>
'4f:0c:d6:b7:f2:43:3c:08:df:62:e3:b2:27:2e:3c:7a');
END;
/
```

After creating the credential object, it displays in the `dba_credentials` table:

```
SELECT owner, credential_name
FROM dba_credentials
WHERE credential_name LIKE '%NATIVE%';
```

```
OWNER CREDENTIAL_NAME
-----
ADMIN OCI_NATIVE_CRED
```

Example: Using User Principals

In this example, user1 is a member of the group adb-admins. All members of this group are given permission to manage all data catalogs in mycompartment, and to read from object-store in mycompartment.

1. Allow users that are members of adb-admins to manage all data catalogs within mycompartment.

```
allow group adb-admins to manage data-catalog-family in compartment
mycompartment
```

2. Allow users that are members of adb-admins to read any object in any bucket within mycompartment.

```
allow group adb-admins to read objects in compartment mycompartment
```

Connection Management Subprograms

This table lists the DBMS_DCAT package procedures used to create, query and drop Data Catalog connections.

Subprogram	Description
SET_DATA_CATALOG_CREDENTIAL Procedure	Set the data catalog access credential used by a specific connection to the data catalog
SET_OBJECT_STORE_CREDENTIAL Procedure	Set the credential used by the given unique connection identifier for accessing the Object Store
SET_DATA_CATALOG_CONNECTION Procedure	Create a connection to the given data catalog
UNSET_DATA_CATALOG_CONNECTION Procedure	Remove an existing Data Catalog connection

SET_DATA_CATALOG_CREDENTIAL Procedure

This procedure sets the Data Catalog access credential used by a specific connection to the Data Catalog.

Syntax:

```
PROCEDURE DBMS_DCAT.SET_DATA_CATALOG_CREDENTIAL(
    credential_name VARCHAR2(128) DEFAULT NULL,
    dcat_con_id      VARCHAR2 DEFAULT NULL
);
```

Parameters:

Parameter	Description
credential_name	The credential used for accessing the Data Catalog.

Parameter	Description
dcat_con_id	The unique Data Catalog connection identifier. This credential is used for the connection identified by dcat_con_id. The default is Null.

This credential must have Manage Data Catalog permissions; see [Data Catalog Policies](#).

SET_OBJECT_STORE_CREDENTIAL Procedure

This procedure sets the credential that is used by the given unique connection identifier for accessing the Object Store. Changing the Object Store access credential alters all existing synced tables to use the new credential.

Syntax:

```
PROCEDURE DBMS_DCAT.SET_OBJECT_STORE_CREDENTIAL(  
    credential_name VARCHAR2(128),  
    dcat_con_id     IN VARCHAR2 DEFAULT NULL  
);
```

Parameters:

Parameter	Description
credential_name	The credential used by the external tables for accessing the Object Store.
dcat_con_id	The unique Data Catalog connection identifier. The default is NULL.

SET_DATA_CATALOG_CONN Procedure

This procedure creates a connection to the given Data Catalog. The connection is required to synchronize metadata with Data Catalog. An Autonomous AI Database instance can connect to multiple Data Catalog instances and supports connecting to OCI Data Catalogs and AWS Glue Data Catalogs.

Syntax:

```
PROCEDURE DBMS_DCAT.SET_DATA_CATALOG_CONN (  
    region          VARCHAR2 DEFAULT NULL,  
    endpoint        VARCHAR2 DEFAULT NULL,  
    catalog_id      VARCHAR2 DEFAULT NULL,  
    dcat_con_id     VARCHAR2 DEFAULT NULL,  
    catalog_type    VARCHAR2 DEFAULT NULL,  
    token_endpoint  VARCHAR2 DEFAULT Null  
);
```

Parameters:

Parameter	Description
region	The Data Catalog region. If the endpoint is specified, region is optional. If both endpoint and region are given, then the endpoint takes precedence. Default is NULL.
endpoint	The Data Catalog endpoint. If the region is specified, endpoint is optional. If both endpoint and region are given, then the endpoint takes precedence. Default is NULL.

Parameter	Description
<code>catalog_id</code>	The unique Oracle Cloud Identifier (OCID) for the Data Catalog instance. When connecting to AWS Glue Data Catalogs, <code>catalog_id</code> is optional.
<code>dcat_con_id</code>	A unique Data Catalog connection identifier. This identifier is required when connecting to multiple Data Catalogs and is optional when connecting to only one. It is used to refer to the Data Catalog connection in subsequent calls or when querying views. If no identifier is specified this procedure generates a NULL connection identifier. The following restrictions apply for <code>dcat_con_id</code> : • It must be unique within the Autonomous AI Database instance. • It must start with a letter. • It may contain alphanumeric characters, underscores (<code>_</code>), dollar signs (<code>\$</code>), and pound signs (<code>#</code>). • It must be at least 16 characters long.
<code>catalog_type</code>	The type of data catalog to connect. Allowed values: • <code>OCI_DCAT</code> - OCI Data Catalog • <code>AWS_GLUE</code> - AWS Glue Data Catalog • <code>NULL</code> - The catalog type is automatically detected from the provided region or endpoint.
<code>token_endpoint</code>	The access token endpoint for OAuth authentication. Default is <code>NULL</code> .

Usage:

You only need to call this procedure once to set the connection. As part of the connection process, Autonomous AI Database adds custom properties to Data Catalog. These custom properties are accessible to Data Catalog users and allow you to override default names (for schemas, tables and columns) and column data types.

Before creating a connection, credentials must be created and set. For a description of the connection process, see [Typical Workflow with Data Catalog for OCI Data Catalogs](#) and [User Workflow for Querying with AWS Glue Data Catalog for AWS Glue Data Catalogs](#).

Example: Connecting with a known OCID:

In this example, Autonomous AI Database is connecting to Data Catalog in the `uk-london-1` region. The `catalog_id` parameter uses the Oracle Cloud Identifier (`ocid`) for the Data Catalog instance. The type of Data Catalog is automatically determined: AWS Glue Data Catalog or OCI Data Catalog.

```
BEGIN
  DBMS_DCAT.SET_DATA_CATALOG_CONN(
    region=>'uk-london-1',
    catalog_id=>'ocid1.datacatalog.oc1.uk-london-1...');
END;
/
```

Example: Connecting to an AWS Glue Data Catalog

A connection is the association between an Autonomous AI Database instance and an AWS Glue Data Catalog. After a successful connection, the Autonomous AI Database instance is able to synchronize with AWS Glue. Each AWS account has one AWS Glue Data Catalog per region and each catalog can be accessed using the corresponding service endpoint for each region. An Autonomous AI Database instance can be associated with an AWS Glue Data Catalog by invoking the API `DBMS_DCAT.SET_DATA_CATALOG_CONN` and specify the endpoint for the region where the catalog resides.

See [AWS Glue endpoints and quotas](#).

In this example, Autonomous AI Database is connecting to an AWS Glue Data Catalog in the uk-london-1 region. Because this is an AWS Glue Data Catalog connection, the `catalog_id` parameter is not needed.

```
BEGIN
  DBMS_DCAT.SET_DATA_CATALOG_CONN(
    region=>'uk-london-1',
    catalog_type=>'AWS_GLUE'
  );
END;
```

UNSET_DATA_CATALOG_CONN Procedure

This procedure removes an existing Data Catalog connection.

Syntax:

Note

Invoking this procedure drops all of the protected schemas and external tables that were created as part of previous synchronizations. It does not impact the metadata in Data Catalog.

```
PROCEDURE DBMS_DCAT.UNSET_DATA_CATALOG_CONN (
  dcat_con_id IN VARCHAR2 DEFAULT NULL
);
```

Parameters:

Parameter	Description
<code>dcat_con_id</code>	The unique Data Catalog connection identifier. Default is Null.

Synchronization Subprograms

Running a synchronization, creating and dropping a synchronization job, and dropping synchronized schemas can be performed with the procedures listed in this table.

Note

On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the `DBMS_DCAT.RUN_SYNC` procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See [DROP_SYNC_JOB Procedure](#) and [CREATE_SYNC_JOB Procedure](#).

Subprogram	Description
CREATE_SYNC_JOB Procedure	Create a scheduler job to invoke <code>RUN_SYNC</code> periodically

Subprogram	Description
DROP_SYNC_JOB Procedure	Drop an existing sync job for the given unique connection identifier
DROP_SYNCED_SCHEMA S Procedure	Drop all previously synchronized schemas for the given unique connection identifier
RUN_SYNC Procedure	Run a synchronization operation

RUN_SYNC Procedure

This procedure runs a synchronization operation and is the entry point to the synchronization. As an input, it takes lists of selected data catalog assets, folders and entities and materializes them by creating, dropping, and altering external tables.

The `sync_option` parameter specifies which operation the `RUN_SYNC` procedure performs: `SYNC`, `DELETE` or `REPLACE`. The operation is performed over entities within the scope of the `synced_objects` parameter.

Every call to the `RUN_SYNC` procedure returns a unique `operation_id` that can be used to query the `USER_LOAD_OPERATIONS` view to obtain information about the status of the sync and the corresponding `log_table`. The `DBMS_DCAT$SYNC_LOG` view can be queried for easy access to the `log_table` for the last sync operation executed by the current user. For further details, see `DBMS_DCAT$SYNC_LOG` View.

Note

On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the `DBMS_DCAT.RUN_SYNC` procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See [DROP_SYNC_JOB Procedure](#) and [CREATE_SYNC_JOB Procedure](#).

Synchronizing Partitioned Logical Entities or Glue Tables

The `RUN_SYNC` procedure creates a partitioned external table for each logical entity or Glue table when all three of the following apply:

1. The OCI data catalog logical entity or Glue table has one or more partitioned attributes.
2. For OCI data catalogs, the logical entity is derived from a prefix-based filename pattern. Partitioned logical entities derived from regex-based patterns are not supported.
3. For OCI data catalogs, the logical entity is based on partitioned data that follows the hive-style or non-hive folder format. Logical entities based on partitioned data that follow the non-hive style format using object names are not supported.
 - Example 1. Logical entities based on harvested objects that follow the Hive style partitioning format with prefix-based filename patterns.

Consider the following objects:

```
Bucket: MYBUCKET
cluster1/db1.db/sales/country=USA/year=2020/month=01/sales1.csv
cluster1/db1.db/sales/country=USA/year=2020/month=01/sales2.csv
cluster1/db1.db/sales/country=USA/year=2020/month=02/sales1.csv
```


Harvesting the bucket using a filename pattern with a starting folder prefix of cluster1/db1.db generates a logical entity named SALES with three partition attributes: country, year, and month. The type for partitioned attributes is Partition while the type for non-partitioned attributes is Primitive.

- Example 2. Logical entities based on harvested objects that follow the non-Hive style partitioning format with prefix-based filename patterns. Consider the following objects:

```
Bucket: MYBUCKET
cluster2/db2.db/sales/USA/2020/01/sales1.csv
cluster2/db2.db/sales/USA/2020/01/sales2.csv
cluster2/db2.db/sales/USA/2020/02/sales1.csv
```

Harvesting the bucket using a filename pattern with a starting folder prefix of cluster2/db2.db generates a logical entity named SALES with three partition attributes: name0, name1, and name2. The only difference between the generated logical entity compared to Example 1, is that the names of partitioned attributes are auto generated, while in Example 1 they are extracted from the URL (country, year, and month respectively).

For a complete end to end example of synchronizing partitioned logical entities, see Example: A Partitioned Data Scenario.

Incremental Synchronization of Partitioned Logical Entities/Glue Tables

Every call to the RUN_SYNC procedure specifies a set of OCI data catalog logical entities or AWS Glue tables to be synced with the database. When a logical entity or Glue table is present in two RUN_SYNC calls, the second call preserves and possibly alters existing external tables. The following table shows which logical entity or Glue table changes are supported when the logical entity or Glue table is partitioned:

Logical Entity or Glue Table Change	Action
Addition, removal, or update of a partition	All partitions of the external partitioned table are updated, regardless of whether a change has been detected by the data catalog.
Addition of a partitioned attribute	Adding a partitioned column to an external partitioned table is not supported. An exception is raised.
Deletion of a partition attribute	Dropping a partitioned column from an external partitioned table is not supported. An exception is raised.
Renaming of a partitioned attribute	Renaming a partitioned column in an external partitioned table is not supported. An exception is raised.

Syntax:

```
PROCEDURE DBMS_DCAT.RUN_SYNC (
    synced_objects IN CLOB,
    sync_option    IN VARCHAR2 DEFAULT 'SYNC',
    error_semantics IN VARCHAR2 DEFAULT 'SKIP_ERRORS',
    log_level      IN VARCHAR2 DEFAULT 'INFO',
    grant_read     IN VARCHAR2 DEFAULT NULL,
    dcat_con_id    IN VARCHAR2 DEFAULT NULL
);
PROCEDURE DBMS_DCAT.RUN_SYNC (
    synced_objects IN CLOB,
```

```

sync_option      IN VARCHAR2 DEFAULT 'SYNC',
error_semantics  IN VARCHAR2 DEFAULT 'SKIP_ERRORS',
log_level        IN VARCHAR2 DEFAULT 'INFO',
grant_read       IN VARCHAR2 DEFAULT NULL,
operation_id     OUT NOCOPY NUMBER,
dcat_con_id      IN VARCHAR2 DEFAULT NULL
);

```

Parameters:

Parameter	Description
synced_objects	This parameter is a JSON document that specifies the data catalog objects to synchronize. For OCI Data Catalogs, the JSON document specifies a set of entities in multiple granularity: data assets, folders (Object Store buckets) or logical entities. It contains an <code>asset_list</code> that is either an array of asset objects or an array containing a single "*" string that stands for 'sync all (object store) data assets in the catalog'. For AWS Glue Data Catalogs, the JSON document specifies a list of tables in multiple granularity: databases, tables. The document specifies a list of databases. Users can restrict the set of tables to be synced by specifying individual tables within a database.
sync_option	(Optional) There are three options: • SYNC (Default) - This option ensures that what is in the data catalog, over the <code>synced_objects</code> scope, is represented in the Autonomous AI Database. If a logical entity or Glue table was deleted from the data catalog, since the last sync operation, then it is deleted in the Autonomous AI Database. The following operations are performed over the <code>synced_objects</code> scope: – Adds tables for new data catalog entities – Removes tables for deleted data catalog entities – Updates properties (such as name, columns and data types) for existing tables • DELETE - Deletes tables within the <code>synced_objects</code> scope. • REPLACE - Replaces all currently synced objects with the objects within the <code>synced_objects</code> scope.
error_semantics	(Optional) This parameter specifies the error behavior. If set to <code>SKIP_ERRORS</code>, the sync attempts to continue despite errors encountered for individual entities. If set to <code>STOP_ON_ERROR</code>, the procedure fails on the first encountered error. The default is <code>SKIP_ERRORS</code>.
log_level	(Optional) This parameter specifies the following values in increasing level of logging detail: <code>OFF</code>, <code>FATAL</code>, <code>ERROR</code>, <code>WARN</code>, <code>INFO</code>, <code>DEBUG</code>, <code>TRACE</code>, <code>ALL</code>. The default is <code>INFO</code>.
grant_read	(Optional) This parameter is a list of users/roles that are automatically granted <code>READ</code> privileges on all external tables processed by this invocation of <code>RUN_SYNC</code>. All users/roles in the <code>grant_read</code> list are given <code>READ</code> privileges on all new or already existing external tables that correspond to entities specified by the <code>synced_objects</code> parameter. The <code>RUN_SYNC</code> procedure preserves already granted privileges on synced external tables.
operation_id	(Optional) This parameter is used to find the corresponding entry in <code>USER_LOAD_OPERATIONS</code> for the sync and determine the name of the log table. Note: A version of <code>RUN_SYNC</code> that does not return an <code>operation_id</code> is available so users can query <code>USER_LOAD_OPERATIONS</code> for the latest sync.

Parameter	Description
dcat_con_id	This parameter is the unique data catalog connection identifier that was specified when the connection to the data catalog was created. See SET_DATA_CATALOG_CONN Procedure . This parameter identifies which connection is used for synchronization and becomes a part of the derived schema name. See Synchronization Mapping for a description of how the schema name is derived. The parameter default is NULL.

Example: Synchronize all OCI Data Catalog entities

In the following example, all Data Catalog entities are synchronized.

```
EXEC DBMS_DCAT.RUN_SYNC(synced_objects=>'{"asset_list":["*"]}');
```

Example: synced_objects parameter for synchronizing all OCI Data Catalog data assets

The following is an example synced_objects parameter for synchronizing all (Object Storage) data assets in the Data Catalog.

```
{"asset_list" : ["*"]}
```

Example: synced_objects parameter for synchronizing specific OCI Data Catalog data assets

The following is an example synced_objects parameter for synchronizing two data assets.

```
{"asset_list": [
  {
    "asset_id": "...-9e2e7f78fd5f"
  },
  {
    "asset_id": "...-9e2e7f74523"
  }
]}
```

Example: synced_objects parameter for synchronizing specific OCI Data Catalog entities within a data asset

The following shows an example synced_objects parameter for synchronizing two entities within the data asset.

```
{"asset_list": [
  {
    "asset_id": "...-9e2e7f78fd5f",
    "folder_list": [
      "f1",
      "f2"
    ]
  }
]}
```

Example: synced_objects parameter for synchronizing specific OCI Data Catalog folders and entities within a data asset

The following shows an example `synced_objects` parameter for synchronizing two folders and two entities within the data asset.

```
{
  "asset_list": [
    {
      "asset_id": "...-9e2e7f78fd5f",
      "entity_list": [
        "entity1",
        "entity2"
      ],
      "folder_list": [
        "f1",
        "f2"
      ]
    }
  ]
}
```

Example: `synced_objects` Parameter for Synchronizing All AWS Glue Data Catalog Databases

The following shows an example `synced_objects` parameter for synchronizing all databases in the AWS Glue Data Catalog.

```
{
  "database_list": ["*"]
}
```

Example: `synced_objects` Parameter for Synchronizing Two AWS Glue Data Catalog Databases

The following shows an example `synced_objects` parameter for synchronizing two AWS Glue Data Catalog databases.

```
{
  "database_list": [
    { "database": "tpcdscsv" },
    { "database": "tpcdsparquet" }
  ]
}
```

Example: `synced_objects` Parameter for Synchronizing Three AWS Glue Data Catalog Databases

The following shows an example `synced_objects` parameter for synchronizing three tables from an AWS Glue Data Catalog database.

```
{
  "database_list": [
    {
      "database": "tpcdsparquet",
      "table_list": [
        "tpcdsparquet_customer",
        "tpcdsparquet_item",
        "tpcdsparquet_web_sales"
      ]
    }
  ]
}
```

CREATE_SYNC_JOB Procedure

This procedure creates a scheduler job to invoke `RUN_SYNC` periodically

It takes as input the set of objects to be synced, the error semantics, the log level, and a repeat interval. See [RUN_SYNC Procedure](#) for further details on how synchronization works.

There can only be a single sync job. The `CREATE_SYNC_JOB` procedure fails if another job is already specified, unless the `force` parameter is set to `TRUE`. If `force` is set to `TRUE` the previous job is dropped.

If a scheduler job attempts to run while another sync is in progress, the scheduler job fails.

Note

On April 4, 2022, the `sync_option` and `grant_read` parameters were added to the `DBMS_DCAT.RUN_SYNC` procedure. To ensure correct performance of scheduled sync jobs created prior to that date, you need to drop and recreate the scheduled sync jobs. See [DROP_SYNC_JOB Procedure](#) and [CREATE_SYNC_JOB Procedure](#).

Syntax:

```
PROCEDURE DBMS_DCAT.CREATE_SYNC_JOB (
    synced_objects    IN CLOB,
    error_semantics   IN VARCHAR2 DEFAULT 'SKIP_ERRORS',
    log_level         IN VARCHAR2 DEFAULT 'INFO',
    repeat_interval   IN VARCHAR2,
    force             IN VARCHAR2 DEFAULT 'FALSE',
    grant_read        IN VARCHAR2 DEFAULT NULL,
    sync_option       IN VARCHAR2 DEFAULT 'SYNC',
    dcat_con_id       IN VARCHAR2 DEFAULT NULL
);
```

Parameters:

Parameter	Description
<code>synced_objects</code>	A JSON object specifying the objects to be synced, as described in the <code>RUN_SYNC</code> procedure. See RUN_SYNC Procedure .
<code>error_semantics</code>	(Optional) Error behavior, as specified for <code>RUN_SYNC</code> . Default is <code>SKIP_ERRORS</code> .
<code>log_level</code>	(Optional) Logging level, as specified for <code>RUN_SYNC</code> . Default is <code>INFO</code> .
<code>repeat_interval</code>	Repeat interval for the job, with the same semantics as the <code>repeat_interval</code> parameter of the <code>DBMS_SCHEDULER.CREATE_JOB</code> procedure. For details on the <code>repeat_interval</code> , see Overview of Creating Jobs .
<code>force</code>	(Optional) If <code>TRUE</code> , existing sync jobs are deleted first. If <code>FALSE</code> , the <code>CREATE_SYNC_JOB</code> procedure fails if a sync job already exists. Default is <code>FALSE</code> .
<code>grant_read</code>	(Optional) List of users/roles to be granted <code>READ</code> on the synced external tables, as described for procedure <code>RUN_SYNC</code> . See RUN_SYNC Procedure .
<code>sync_option</code>	(Optional) Behavior with respect to entities that have already been synced through a previous <code>RUN_SYNC</code> operation, as described for procedure <code>RUN_SYNC</code> . See RUN_SYNC Procedure .
<code>dcat_con_id</code>	This parameter is the unique Data Catalog connection identifier that was specified when the connection to Data Catalog was created. See SET_DATA_CATALOG_CONN Procedure . This parameter identifies which connection is used for synchronization and becomes a part of the derived schema name. See Synchronization Mapping for a description of how the schema name is derived. The parameter default is <code>NULL</code> .

DROP_SYNC_JOB Procedure

This procedure drops an existing sync job for the given unique connection identifier.

Syntax:

```
PROCEDURE DBMS_DCAT.DROP_SYNC_JOB (  
    dcat_con_id IN VARCHAR2 DEFAULT NULL  
);
```

Parameters:

Parameter	Description
dcat_con_id	The unique Data Catalog connection identifier. The default is NULL.

DROP_SYNCED_SCHEMAS Procedure

This procedure drops all previously synchronized schemas for the given unique connection identifier.

Syntax:

```
PROCEDURE DBMS_DCAT.DROP_SYNCED_SCHEMAS (  
    dcat_con_id IN VARCHAR2 DEFAULT NULL  
);
```

Parameters:

Parameter	Description
dcat_con_id	The unique Data Catalog connection identifier. The default is NULL.

CS_SESSION Package

The CS_SESSION package provides an interface to switch the database service and consumer group of the existing session.

Note

Support for the CS_SESSION package is available in Oracle Database 19c starting with version 19.28, and in Oracle Database 26ai starting with version 23.9.

When a connection is established with an Autonomous AI Database, that session is assigned a consumer group. For example, a session could be created using a connection to the LOW service of an Autonomous AI Database. You might want to switch the consumer group, for example from LOW to HIGH. The CS_SESSION package provides an API for switching. See *Predefined Database Service Names for Autonomous AI Databases* for more information.

The consumer groups determine concurrency and degree of parallelism (DOP). For example, statements on a connection established to the LOW database service run serially. Statements on a connection established to the HIGH database service run in parallel. If you have a

workload that requires serial statement processing with switching to a HIGH consumer group for a few statements, the CS_SESSION package enables you to switch.

SWITCH_SERVICE Procedure

This procedure switches the database service and consumer group of the current session.

Syntax

```
CS_SESSION.SWITCH_SERVICE(service_name IN varchar2);
```

Parameters

Parameter	Description
service_name	Specifies the consumer group to update. Depending on the workload, valid values are: HIGH, MEDIUM, LOW, TP, or TPURGENT.

Usage Notes

When called, the procedure switches the session to the specified service and the related consumer group. If the specified service does not exist in that database, an error message is provided. For example, if you specify 'TP' as the service name on a Lakehouse workload, the error indicates it's not a valid service name. No error is reported if the current service and the specified service are identical.

The procedure does not reset session attributes. Anything the user set for their session before calling this procedure will continue as-is. For example, if a session parameter was modified and then later the session switched to a different service, the parameter value will stay the same.

Example

```
BEGIN
  CS_SESSION.SWITCH_SERVICE('HIGH');
END;
/
```

Security and Access

The ADMIN user is granted EXECUTE privilege on CS_SESSION with GRANT OPTION. The privilege is also granted to DWROLE without the GRANT OPTION.

Additional Security Considerations

If a user is granted EXECUTE privileges on this procedure and you do not want that user to switch to a specific service, you can use a AFTER SET CONTAINER trigger to block the operation. This is achieved by creating an AFTER SET CONTAINER trigger.

```
CREATE OR REPLACE TRIGGER SESS_SWITCH
AFTER SET CONTAINER ON DATABASE
BEGIN
  IF SYS_CONTEXT('USERENV', 'SESSION_USER') = 'USER' and
     SYS_CONTEXT('USERENV', 'SERVICE_NAME') =
     'serviceexample_low.adwc.oraclecloud.com'
```

```

THEN
    NULL;
ELSE
    RAISE_APPLICATION_ERROR(-20001, 'Denied! You are not allowed to switch
service in the database');
END IF;
END;
/

```

Error Messages

The following table describes exceptions for CS_SESSION

Error	Message	Cause
20001	Invalid service name. Valid values are HIGH, MEDIUM, LOW.	For a data warehouse workload, a value other than 'HIGH', 'MEDIUM', 'LOW' was specified.
20001	Invalid service name. Valid values are HIGH, MEDIUM, LOW, TP, TPURGENT.	For a transaction processing workload, a value other than 'HIGH', 'MEDIUM', 'LOW', 'TP', 'TPURGENT' was specified.
20002	Service switch failed.	Failed to switch to the new service.

Summary of DBMS_CLOUD_ADMIN Subprograms

This section covers the DBMS_CLOUD_ADMIN subprograms provided with Autonomous AI Database.

Subprogram	Description
ATTACH_FILE_SYSTEM Procedure	This procedure attaches a file system in a directory on your database.
CREATE_DATABASE_LINK Procedure	This procedure creates a database link to a target database. There are options to create a database link to another Autonomous AI Database instance, to an Oracle Database that is not an Autonomous AI Database, or to a non-Oracle Database using Oracle-managed heterogeneous connectivity.
DETACH_FILE_SYSTEM Procedure	This procedure detaches a file system from a directory on your database.
DISABLE_EXTERNAL_AUTHENTICATION Procedure	This procedure disables external authentication for the Autonomous AI Database instance.
DROP_DATABASE_LINK Procedure	This procedure drops a database link.
ENABLE_EXTERNAL_AUTHENTICATION Procedure	This procedure enables a user to logon to Autonomous AI Database using the specified external authentication scheme.

ATTACH_FILE_SYSTEM Procedure

This procedure attaches a file system in the database.

The DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM procedure attaches a file system in your database and stores information about the file system in the DBA_CLOUD_FILE_SYSTEMS view.

Syntax

```

DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name      IN VARCHAR2,
    file_system_location  IN VARCHAR2,
    directory_name        IN VARCHAR2,
    description           IN VARCHAR2 DEFAULT NULL,
    params               IN CLOB DEFAULT NULL
);

```

Parameters

Parameter	Description
file_system_name	Specifies the name of the file system. This parameter is mandatory.
file_system_location	Specifies the location of the file system. The value you supply with file_system_location consists of a Fully Qualified Domain Name (FQDN) and a file path in the form: <i>FQDN:file_path</i> .

Note

You can use an FQDN or an IP address.

For example:

- FQDN: myhost.sub000445.myvcn.oraclevcn.com
For Oracle Cloud Infrastructure File Storage set the FQDN in **Show Advanced Options** when you create a file system. See [Creating File Systems](#) for more information.
- File Path: /results
This parameter is mandatory.

directory_name	Specifies the directory name for the attached file system. The directory must exist. This parameter is mandatory.
description	(Optional) Provides a description of the task.
params	A JSON string that provides an additional parameter for the file system. nfs_version: Specifies the NFS version to use when NFS is attached (NFSv3 or NFSv4). Valid values: 3, 4. Default value: 3

Examples:

Attach to an NFSv3 file system:

```

BEGIN
    DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
        file_system_name      => 'FSS',
        file_system_location  => 'myhost.sub000445.myvcn.oraclevcn.com:/results',
        directory_name        => 'FSS_DIR',
        description           => 'Source NFS for sales data'
    );

```

```
END;
/
```

Attach to an NFSv4 file system:

```
BEGIN
  DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (
    file_system_name      => 'FSS',
    file_system_location  => 'myhost.sub000445.myvcn.oraclevcn.com:/results',
    directory_name        => 'FSS_DIR',
    description           => 'Source NFS for sales data',
    params                => JSON_OBJECT('nfs_version' value 4)
  );
END;
/
```

Usage Notes

- To run this procedure you must be logged in as the ADMIN user or have the EXECUTE privilege on DBMS_CLOUD_ADMIN.
- You must have WRITE privilege on the directory object in the database to attach a file system using DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM.
- The DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM procedure can only attach a private File Storage Service in databases with Private Endpoints enabled.
See [OCI File Storage Service](#) and Configure a Development System to Access the Database for more information.
- The DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM procedure looks up the Network File System hostname on the customer's virtual cloud network (VCN). The error "ORA-20000: Mounting NFS fails" is returned if the hostname specified in the location cannot be located.
- Oracle Cloud Infrastructure File Storage uses NFSv3 to share
- If you attach to non-Oracle Cloud Infrastructure File Storage systems, the procedure supports NFSv3 and NFSv4
- If you have an attached NFS server that uses NFSv3 and the NFS version is updated to NFSv4 in the NFS server, you must run DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM and then DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM (using the params parameter with nfs_version set to 4). This attaches NFS with the matching protocol so that Autonomous AI Database can access the NFSv4 Server. Without detaching and then reattaching, the NFS server will be inaccessible and you may see an error such as: "Protocol not supported".

CREATE_DATABASE_LINK Procedure

This procedure creates a database link to a target database in the schema calling the API.

The overloaded forms support the following:

- When you use the gateway_params parameter, this enables you to create a database link with Oracle-managed heterogeneous connectivity where the link is to a supported non-Oracle database.
- When you use the rac_hostnames parameter, this enables you to create a database link from an Autonomous AI Database on a private endpoint to a target Oracle RAC database.

In this case, you use the `rac_hostnames` parameter to specify the host names of one or more individual nodes of the target Oracle RAC database.

Syntax

```
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name      IN VARCHAR2,
    hostname           IN VARCHAR2,
    port               IN NUMBER,
    service_name       IN VARCHAR2,
    ssl_server_cert_dn IN VARCHAR2 DEFAULT,
    credential_name    IN VARCHAR2 DEFAULT,
    directory_name     IN VARCHAR2 DEFAULT,
    gateway_link       IN BOOLEAN DEFAULT,
    public_link        IN BOOLEAN DEFAULT,
    private_target     IN BOOLEAN DEFAULT,
    gateway_params     IN CLOB DEFAULT);
```

```
DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name      IN VARCHAR2,
    rac_hostnames     IN CLOB,
    port               IN NUMBER,
    service_name       IN VARCHAR2,
    ssl_server_cert_dn IN VARCHAR2 DEFAULT,
    credential_name    IN VARCHAR2 DEFAULT,
    directory_name     IN VARCHAR2 DEFAULT,
    gateway_link       IN BOOLEAN DEFAULT,
    public_link        IN BOOLEAN DEFAULT,
    private_target     IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
<code>db_link_name</code>	The name of the database link to create.
<code>hostname</code>	The hostname for the target database. Specifying <code>localhost</code> for <code>hostname</code> as is not allowed. When you specify a connection with Oracle-managed heterogeneous connectivity by supplying the <code>gateway_params</code> parameter, note the following: - When the <code>db_type</code> value is <code>google_bigquery</code> the <code>hostname</code> is not used and you can provide value such as <code>example.com</code>. - When the <code>db_type</code> value is <code>snowflake</code> the <code>hostname</code> is the Snowflake account identifier. To find your Snowflake account identifier, see Account Identifier Formats by Cloud Platform and Region. Use this parameter or <code>rac_hostnames</code>, do not use both. The <code>DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK</code> input should mention the scan name of the target Dedicated Autonomous AI Database as the 'hostname' parameter.

Parameter	Description
rac_hostnames	Specifies hostnames for the target Oracle RAC database. The value is a JSON array that specifies one or more individual host names for the nodes of the target Oracle RAC database. Multiple host names can be passed in JSON, separated by a ", ". For example: <pre>'["sales1-svr1.domain", "sales1-svr2.domain", "sales1-svr3.domain"]'</pre> When the target is an Oracle RAC database, use the rac_hostnames parameter to specify one or more hostnames with DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK. This allows you to take advantage of the high availability capabilities of Oracle RAC. Using an IP address, a SCAN IP, or a SCAN hostname in the rac_hostnames value is not supported. When you specify a list of host names in the rac_hostnames parameter, CREATE_DATABASE_LINK uses all of the specified host names as addresses in the connect string. If one of the specified hosts is not available on the target Oracle RAC database, Autonomous Database automatically attempts to connect using another host name from the list. Use this parameter or hostname, do not use both. Specifying localhost for a rac_hostname value is not allowed.
port	Specifies the port for the connections to the target database. When you specify a connection with Oracle-managed heterogeneous connectivity using the gateway_params parameter, set the port based on the db_type value: • awsredshift: use port 5439 • azure: use port 1433 • db2: use port 2500 for Db2 versions >= 11.5.6 • db2: use port 5000 for Db2 versions <= 11.5.5 • google_analytics: use port 443 • google_bigquery: use port 443 • hive: use port 433 • mongodb: use port 27017 • mysql: use port 3306 • mysql_community: use port 3306 • postgres: use port 5432 • salesforce: use port 19937 • servicenow: use port 443 • snowflake: use port 443

Parameter	Description
service_name	The service_name for the database to link to. For a target Autonomous AI Database, find the service name by one of the following methods: - Look in the <code>tnsnames.ora</code> file in the <code>wallet.zip</code> that you download from an Autonomous AI Database for your connection. - Click Database connection on the Oracle Cloud Infrastructure Console. Each connection string listed in the Connection string column includes a <code>service_name</code> entry with the connection string for the corresponding service. See View Connection Strings for an Autonomous AI Database for more information. - Query <code>V\$SERVICES</code> view. For example: <pre>SELECT name FROM V\$SERVICES;</pre> When you specify a connection with Oracle-managed heterogeneous connectivity using the <code>gateway_params</code> parameter, the <code>service_name</code> is the database name of the non-Oracle database.
ssl_server_cert_dn	The DN value found in the server certificate. Oracle-managed heterogeneous connectivity is preconfigured with a wallet that contains most of the common trusted root and intermediate SSL certificates. The <code>ssl_server_cert_dn</code> must be NULL when you supply the <code>gateway_params</code> parameter or do not include the <code>ssl_server_cert_dn</code> parameter (the default value is NULL). Public Endpoint Link to an Autonomous AI Database Target without a Wallet: To connect to an Autonomous AI Database target on a public endpoint without a wallet (TLS): - The <code>directory_name</code> parameter must be NULL. - The <code>ssl_server_cert_dn</code> parameter must be NULL or do not include this parameter (the default value is NULL). Private Endpoint Link without a Wallet: To connect to an Oracle Database on a private endpoint without a wallet: - The target database must be on a private endpoint. - The <code>directory_name</code> parameter must be NULL. - The <code>ssl_server_cert_dn</code> parameter must be NULL or do not include this parameter (the default is NULL). - The <code>private_target</code> parameter must be TRUE.
credential_name	The name of a stored credential created with <code>DBMS_CLOUD.CREATE_CREDENTIAL</code>. This is the credentials to access the target database.

Parameter	Description
directory_name	The directory for the <code>cwallet.sso</code> file. The default value for this parameter is <code>'data_pump_dir'</code>. Oracle-managed heterogeneous connectivity is preconfigured with a wallet that contains most of the common trusted root and intermediate SSL certificates. The <code>directory_name</code> parameter is not required when you supply the <code>gateway_params</code> parameter. Public Endpoint Link to an Autonomous AI Database Target without a Wallet: To connect to an Autonomous AI Database on a public endpoint without a wallet (TLS): - The <code>directory_name</code> parameter must be NULL. - The <code>ssl_server_cert_dn</code> parameter must be NULL or do not include this parameter (the default value is NULL). In addition, to connect to an Autonomous AI Database with TCP, the <code>ssl_server_cert_dn</code> parameter must be NULL or do not include this parameter (the default value is NULL). Private Endpoint Link without a Wallet: To connect to a target Oracle Database on a private endpoint without a wallet: - The target database must be on a private endpoint. - The <code>directory_name</code> parameter must be NULL. - The <code>ssl_server_cert_dn</code> parameter must be NULL or do not include this parameter (the default value is NULL). - The <code>private_target</code> parameter must be TRUE.
gateway_link	Indicates if the database link is created to another Oracle Database or to an Oracle Database Gateway. If <code>gateway_link</code> is set to FALSE, this specifies a database link to another Autonomous AI Database or to another Oracle Database. If <code>gateway_link</code> is set to TRUE, this specifies a database link to a non-Oracle system. This creates a connect descriptor in the database link definition that specifies (HS=OK). When <code>gateway_link</code> is set to TRUE and <code>gateway_params</code> is NULL, this specifies a database link to a customer-managed Oracle gateway. The default value for this parameter is FALSE.
public_link	Indicates if the database link is created as a public database link. To run <code>DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK</code> with this parameter set to TRUE, the user invoking the procedure must have EXECUTE privilege on the credential associated with the public database link and must have the CREATE PUBLIC DATABASE LINK system privilege. The EXECUTE privilege on the credential can be granted either by the ADMIN user or by the credential owner. The default value for this parameter is FALSE.
private_target	When a database link accesses a hostname that needs to be resolved in a VCN DNS server, specify the <code>private_target</code> parameter with value TRUE. When <code>private_target</code> is TRUE, the <code>hostname</code> parameter must be a single hostname (on a private endpoint, using an IP address, a SCAN IP, or a SCAN hostname is not supported). The default value for this parameter is FALSE.

Parameter	Description
gateway_params	db_type This parameter specifies the target database type for Oracle-managed heterogeneous connectivity to connect to non-Oracle databases. The <code>db_type</code> value is one of: • <code>awsredshift</code> • <code>azure</code> • <code>db2</code> • <code>google_analytics</code> • <code>google_bigquery</code> * See Usage Notes for additional supported gateway_params when <code>db_type</code> is <code>google_bigquery</code>. • <code>hive</code> * See Usage Notes for additional supported gateway_params when <code>db_type</code> is <code>hive</code>. • <code>mongodb</code> • <code>mysql</code> • <code>postgres</code> • <code>salesforce</code> * See Usage Notes for additional supported gateway_params when <code>db_type</code> is <code>salesforce</code>. • <code>servicenow</code> * See Usage Notes for additional supported gateway_params when <code>db_type</code> is <code>servicenow</code>. • <code>snowflake</code> * See Usage Notes for additional supported gateway_params when <code>db_type</code> is <code>snowflake</code>. • <code>NULL</code> When <code>gateway_params</code> is <code>NULL</code> and <code>gateway_link</code> is set to <code>TRUE</code>, this specifies a database link to a customer-managed Oracle gateway. Specify the parameter with the <code>json_object</code> form. For example: <pre>gateway_params => json_object('db_type' value 'awsredshift')</pre> When <code>gateway_params</code> is <code>NULL</code> and <code>private_target</code> is <code>TRUE</code>, if <code>directory_name</code> is <code>NULL</code>, a TCP-based database link is created. When <code>gateway_params</code> is <code>NULL</code> and <code>private_target</code> is <code>TRUE</code>, if <code>directory_name</code> is <code>NULL</code>, a TCPS-based database link is created.

Usage Notes

- When you specify the `gateway_params` parameter, for some `db_type` values, additional gateway_params parameters are supported:

db_type	Additional gateway_params Values
google_analytics	When the <code>db_type</code> is <code>google_analytics</code> , the credential you specify must be a Google OAuth credential (<code>gcp_oauth2</code>) See <code>CREATE_CREDENTIAL</code> Procedure for more information.

db_type	Additional gateway_params Values
google_bigquery	When the db_type is google_bigquery, the credential you specify must be a Google OAuth credential (gcp_oauth2) See CREATE_CREDENTIAL Procedure for more information. When db_type is google_bigquery, the parameter project is valid. This parameter specifies the project name for google_bigquery and is required. The table name you specify when you use SELECT with Google BigQuery must be in quotes. For example: <pre>SELECT * FROM "sales"@GOOGLE_BIGQUERY_LINK</pre>
hive	When db_type is hive, the parameter http_path is valid. This parameter specifies the HttpPath value, if required, to connect to the Hive instance.
salesforce	When the db_type is salesforce, the parameter: security_token is valid. A security token is a case-sensitive alphanumeric code. Supplying a security_token value is required to access Salesforce. For example: <pre>gateway_params => JSON_OBJECT('db_type' value 'salesforce', 'security_token' value 'security_token_value')</pre> See Reset Your Security Token for more information.

db_type	Additional gateway_params Values
servicenow	To connect to ServiceNow and get data you must supply the gateway parameters <code>directory_name</code> and <code>file_name</code>. These parameters specify a model file (REST config file) that maps the JSON response to the relational model. The model file specifies the endpoints, table mapping, and HTTP response code for processing the JSON response. See Model file syntax and Example Model file for more information. When you use <code>gateway_params</code> parameter with <code>db_type</code> <code>servicenow</code>, there are two supported options: – Basic Authentication: you must supply the <code>gateway_params</code> parameter <code>db_type</code> with the value <code>'servicenow'</code>, and supply the <code>directory_name</code> and <code>file_name</code> parameters along with username/password type credentials. – OAuth 2.0 Authentication: you must supply the <code>gateway_params</code> parameter <code>db_type</code> with the value <code>'servicenow'</code>, and the <code>directory_name</code>, <code>file_name</code>, and <code>token_uri</code> parameters, along with OAuth type credentials. The <code>directory_name</code> parameter specifies the directory with the ServiceNow REST config file. You could create this directory as follows: <pre data-bbox="659 800 1284 861">create or replace directory servicenow_dir as 'SERVICENOW_DIR';</pre> Obtain and download the ServiceNow REST config file to the specified directory. For example: <pre data-bbox="659 1005 1336 1100">exec DBMS_CLOUD.get_object('servicenow_dir_cred', 'https://objectstorage.<...>/ servicenow.rest', 'SERVICENOW_DIR');</pre> Set the <code>file_name</code> value to the name of the REST config file you downloaded, <code>"servicenow.rest"</code>. Then you can use the ServiceNow REST config file with either basic authentication or OAuth2.0.
snowflake	When the <code>db_type</code> is <code>SNOWFLAKE</code>, the optional parameters: <code>role</code>, <code>schema</code>, and <code>warehouse</code> are valid. These values specify a different schema, role, or warehouse value, other than the default. For example: <pre data-bbox="659 1415 1114 1570">gateway_params => JSON_OBJECT('db_type' value 'snowflake', 'role' value 'ADMIN', 'schema' value 'PUBLIC', 'warehouse' value 'TEST')</pre>

- When you use the `private_target` parameter, note that database links from an Autonomous AI Database to a database service that is on a private endpoint are only supported in commercial regions and US Government regions.

This feature is enabled by default in all commercial regions.

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

For existing US Government databases on a private endpoint, if you want to create database links from an Autonomous AI Database to a target in a US Government region,

please file a Service Request at [Oracle Cloud Support](#) and request to enable the private endpoint in government regions database linking feature.

US Government regions include the following:

- [Oracle Cloud Infrastructure US Government Cloud with FedRAMP Authorization](#)
- [Oracle Cloud Infrastructure US Federal Cloud with DISA Impact Level 5 Authorization](#)
- When connecting to a non-Oracle database, database linking is only supported if the target database is accessible through a public IP or a public hostname.
- To run `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` with a user other than `ADMIN`, you need to grant `EXECUTE` and `CREATE DATABASE LINK` privileges to that user. For example, run the following command as `ADMIN` to grant privileges to `atpc_user`:

```
GRANT EXECUTE ON DBMS_CLOUD_ADMIN TO atpc_user;
GRANT CREATE DATABASE LINK TO atpc_user;
```

In addition, when you create a Database Link in a schema other than the `ADMIN` schema, for example in a schema named `atpc_user`, the `atpc_user` schema must own the credential you use with `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK`.

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

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

- To create a database link to an Autonomous AI Database, set `GLOBAL_NAMES` to `FALSE` on the source database (non-Autonomous AI Database).

```
SQL> ALTER SYSTEM SET GLOBAL_NAMES = FALSE;
```

System altered.

```
SQL> SHOW PARAMETER GLOBAL_NAMES
NAME                                TYPE          VALUE
-----
global_names                        boolean       FALSE
```

- When the `private_target` parameter is `TRUE`, the `hostname` parameter specifies a private host inside the VCN.

Examples

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DB_LINK_CRED',
    username => 'adb_user',
    password => 'password');
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name => 'SALES LINK',
    hostname => 'adb.eu-frankfurt-1.oraclecloud.com',
    port => '1522',
    service_name => 'example_medium.atpc.example.oraclecloud.com',
```

```

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

```

```

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

```

```

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'PRIVATE_ENDPOINT_CRED',
        username => 'db_user',
        password => 'password'
    );
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'PRIVATE_ENDPOINT_DB_LINK',
        hostname => 'exampleHostname',
        port => '1521',
        service_name => 'exampleServiceName',
        credential_name => 'PRIVATE_ENDPOINT_CRED',
        ssl_server_cert_dn => NULL,
        directory_name => NULL,
        private_target => TRUE);
END;
/

```

```

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'GOOGLE_BIGQUERY_CRED',
        params => JSON_OBJECT( 'gcp_oauth2' value JSON_OBJECT(
            'client_id' value 'client_id',
            'client_secret' value 'client_secret',
            'refresh_token' value 'refresh_token' )));
    DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
        db_link_name => 'GOOGLE_BIGQUERY_LINK',
        hostname => 'example.com',
        port => '443',

```

```

        service_name => 'example_service_name',
        credential_name => 'GOOGLE_BIGQUERY_CRED',
        gateway_params => JSON_OBJECT(
            'db_type' value 'google_bigquery',
            'project' value 'project_name1' ));
END;
/

```

The table name you specify when you use SELECT with Google BigQuery or Google Analytics must be in quotes. For example:

```
SELECT * FROM "sales"@GOOGLE_BIGQUERY_LINK
```

Use the `rac_hostnames` parameter with a target Oracle RAC database on a private endpoint.

```

BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DB_LINK_CRED1',
    username => 'adb_user',
    password => 'password');
  DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK(
    db_link_name => 'SALESLINK',
    rac_hostnames => '["sales1-svr1.example.adb.us-
ashburn-1.oraclecloud.com",
                    "sales1-svr2.example.adb.us-
ashburn-1.oraclecloud.com",
                    "sales1-svr3.example.adb.us-
ashburn-1.oraclecloud.com"]',
    port => '1522',
    service_name => 'example_high.adb.oraclecloud.com',
    ssl_server_cert_dn => 'CN=adb.example.oraclecloud.com,OU=Oracle BMCS
FRANKFURT,O=Oracle Corporation,L=Redwood City,ST=California,C=US',
    credential_name => 'DB_LINK_CRED1',
    directory_name => 'EXAMPLE_WALLET_DIR',
    private_target => TRUE);
END;
/

```

DETACH_FILE_SYSTEM Procedure

This procedure detaches a file system from the database.

The `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure detaches a file system from your database. In addition to that, the `DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM` procedure also removes the information about the file system from the `DBA_CLOUD_FILE_SYSTEMS` view.

Syntax

```

DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM(
    file_system_name          IN VARCHAR2
);

```

Parameters

Parameter	Description
file_system_name	Specifies the name of the file system. This parameter is mandatory.

Example:

```
BEGIN
    DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM (
        file_system_name => 'FSS'
    );
END;
/
```

Usage Notes

- To run this procedure, you must be logged in as the ADMIN user or have the EXECUTE privilege on DBMS_CLOUD_ADMIN.
- You must have the WRITE privilege on the directory object in the database, to detach a file system from a directory using the DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM procedure.
- The DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM procedure can only detach a private File Storage Service in databases with Private Endpoints enabled.
See [OCI File Storage Service](#) and Configure a Development System to Access the Database for more information.
- The DBMS_CLOUD_ADMIN.DETACH_FILE_SYSTEM procedure looks up the Network File System hostname on the customer's virtual cloud network (VCN). The error "ORA-20000: Mounting NFS fails" is returned if the hostname specified in the location cannot be located.

DISABLE_EXTERNAL_AUTHENTICATION Procedure

Disables user authentication with external authentication schemes for the database.

Syntax

```
DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
```

Exceptions

Exception	Error	Description
invalid_ext_auth	ORA-20004	See the accompanying message for a detailed explanation.

Example

```
BEGIN
    DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

PL/SQL procedure successfully completed.

DROP_DATABASE_LINK Procedure

This procedure drops a database link.

Syntax

```
DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK(  
    db_link_name      IN VARCHAR2,  
    public_link       IN BOOLEAN DEFAULT);
```

Parameters

Parameter	Description
db_link_name	The name of the database link to drop.
public_link	To run DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK with public_link set to TRUE, you must have the DROP PUBLIC DATABASE LINK system privilege. The default value for this parameter is FALSE.

Example

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

Usage Notes

After you are done using a database link and you run DBMS_CLOUD_ADMIN.DROP_DATABASE_LINK, to ensure security of your Oracle database remove any stored wallet files. For example:

- Remove the wallet file in Object Store.
- Use DBMS_CLOUD.DELETE_FILE to remove the wallet file from the data_pump_dir directory or from the user defined directory where the wallet file was uploaded.

ENABLE_EXTERNAL_AUTHENTICATION Procedure

Enable users to login to the database with external authentication schemes.

Syntax

```
DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(  
    type      IN VARCHAR2,  
    force     IN BOOLEAN DEFAULT FALSE,  
    params    IN CLOB DEFAULT NULL  
);
```

Parameter

Parameter	Description
type	Specifies the external authentication type. Valid values: or . • 'OCI_IAM' • 'AZURE_AD' • 'CMU' • 'KERBEROS'
force	(Optional) Override a currently enabled external authentication scheme. Valid values are TRUE or FALSE. The default value is FALSE.

Parameter	Description
params	A JSON string that provides additional parameters for the external authentication. CMU parameters: <code>location_uri</code>: specifies the cloud storage URI for the bucket where files required for CMU are stored. If you specify <code>location_uri</code> there is a fixed name directory object <code>CMU_WALLET_DIR</code> created in the database at the path <code>'cmu_wallet'</code> to save the CMU configuration files. In this case, you do not need to supply the <code>directory_name</code> parameter. <code>credential_name</code>: specifies the credentials that are used to download the CMU configuration files from the Object Store to the directory object. Default value is NULL which allows you to provide a Public, Preauthenticated, or pre-signed URL for Object Store bucket or subfolder. <code>directory_name</code>: specifies the directory name where configuration files required for CMU are stored. If <code>directory_name</code> is supplied, you are expected to copy the CMU configuration files <code>dsi.ora</code> and <code>cwallet.sso</code> to this directory object. KERBEROS parameters: <code>location_uri</code>: specifies the cloud storage URI for the bucket where the files required for Kerberos are stored. If you specify <code>location_uri</code> there is a fixed name directory object <code>KERBEROS_DIR</code> created in the database to save the Kerberos configuration files. In this case, you do not need to supply the <code>directory_name</code> parameter. <code>credential_name</code>: specifies the credential that are used to download Kerberos configuration files from the Object Store location to the directory object. Default value is NULL which allows you to provide a Public, Preauthenticated, or pre-signed URL for Object Store bucket or subfolder. <code>directory_name</code>: specifies the directory name where files required for Kerberos are stored. If <code>directory_name</code> is supplied, you are expected to copy the Kerberos configuration files to this directory object. <code>kerberos_service_name</code>: specifies a name to use as the Kerberos service name. This parameter is optional. Default value: When not specified, the <code>kerberos_service_name</code> value is set to the Autonomous AI Database instance's GUID. AZURE_AD parameters: <code>tenant_id</code>: Tenant ID of the Azure Account. Tenant Id specifies the Autonomous AI Database instance's Azure AD application registration. <code>application_id</code>: Azure Application ID created in Azure AD to assign roles/schema mappings for external authentication in the Autonomous AI Database instance. <code>application_id_uri</code>: Unique URI assigned to the Azure Application. This is the identifier for the Autonomous AI Database instance. The name must be domain qualified (this supports cross tenancy resource access). The maximum length for this parameter is 256 characters.

Exceptions

Exception	Error	Description
invalid_ext_auth	ORA-20004	See the accompanying message for a detailed explanation.

Usage Notes

- With type OCI_IAM, if the resource principal is not enabled on the Autonomous AI Database instance, this routine enables resource principal with DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL.
- This procedure sets the system parameters IDENTITY_PROVIDER_TYPE and IDENTITY_PROVIDER_CONFIG to required users to access the instance with Oracle Cloud Infrastructure Identity and Access Management authentication and authorization.

Examples

Enable OCI_IAM Authentication

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'OCI_IAM',
    force=> TRUE );
END;
/
```

PL/SQL procedure successfully completed.

Enable CMU Authentication for Microsoft Active Directory

You pass in a directory name that contains the CMU configuration files through params JSON argument.

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'CMU',
    force => TRUE,
    params => JSON_OBJECT('directory_name' value 'CMU_DIR'); // CMU_DIR
directory object already exists
END;
/
```

PL/SQL procedure successfully completed.

You pass in a location URI pointing to an Object Storage location that contains CMU configuration files through params JSON argument.

```
BEGIN
    DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
        type      => 'CMU',
        params    => JSON_OBJECT('location_uri' value 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
                                'credential_name' value 'my_credential_name')
    );
```

```
END;
/
```

PL/SQL procedure successfully completed.

Enable Azure AD Authentication

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'AZURE_AD',
    force => TRUE,
    params => JSON_OBJECT( 'tenant_id' VALUE '....',
                          'application_id' VALUE '...',
                          'application_id_uri' VALUE '.....' ));
END;
/
```

PL/SQL procedure successfully completed.

Enable Kerberos Authentication

You pass in a directory name that contains Kerberos configuration files through params JSON argument.

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'KERBEROS',
    force => TRUE,
    params => JSON_OBJECT('directory_name' value 'KERBEROS_DIR'); //
KERBEROS_DIR directory object already exists
END;
/
```

PL/SQL procedure successfully completed.

You pass in a location URI pointing to an Object Storage location that contains Kerberos configuration files through params JSON argument:

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'KERBEROS',
    force => TRUE,
    params => JSON_OBJECT('location_uri' value 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
                        'credential_name' value 'my_credential_name');
END;
/
```

You pass in a service name with the `kerberos_service_name` in the params JSON argument:

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
    type => 'KERBEROS',
    force => TRUE,
    params => JSON_OBJECT('directory_name' value 'KERBEROS_DIR', //
KERBEROS_DIR directory object already exists
                        'kerberos_service_name' value 'oracle' ));
```

```
END;
/
```

After Kerberos is enabled on your Autonomous AI Database instance, use the following query to view the Kerberos service name:

```
SELECT SYS_CONTEXT( 'USERENV', 'KERBEROS_SERVICE_NAME' ) FROM DUAL;
```

DBMS_CLOUD_ADMIN Exceptions

The following table describes exceptions for DBMS_CLOUD_ADMIN.

Exception	Code	Description
invalid_service	20001	An invalid service was specified.
service_not_exist	20002	A service specified does not exist.
default_service	20003	A service specified cannot be modified.
invalid_char_set	20029	Missing precondition or invalid (national) character set.
invalid_enc_key_attr	20030	Missing or invalid argument for key management.
Already Using Oracle Managed Key	000000	The Database is already using an Oracle managed key. You are trying to call the procedure while already using an Oracle managed key.
Argument Provided for the procedure	ORA-0000	An argument is provided for the procedure. Expected error message:No arguments required for this procedure.

DBMS_CLOUD_AI Package

The DBMS_CLOUD_AI package, with Select AI, facilitates and configures the translation of natural language prompts to generate, run, explain SQL statements. Also, enables retrieval augmented generation and natural language-based interactions, including chatting with LLMs.

Prerequisites

- [Oracle Public Cloud or Multicloud](#)
- [Exadata Cloud@Customer](#)

Oracle Public Cloud or Multicloud

An outbound connectivity must have been configured using a NAT gateway, by your fleet administrator as described below:

- Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
- After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 1. Go to the **Subnet Details** page for the subnet.

2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** NAT Gateway
 - **Target:** The name of the NAT gateway you just created in the VCNIf such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.
4. Return to the **Subnet Details** page for the subnet.
5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
6. In the side menu, under **Resources**, click **Egress Rules**.
7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** AllIf such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

Summary of DBMS_CLOUD_AI Subprograms

This section covers the DBMS_CLOUD_AI subprograms provided with Autonomous AI Database.

Subprogram	Description
CREATE_PROFILE Procedure	This procedure creates a new AI profile for translating natural language prompts to SQL statements.
Profile Attributes	Provides AI profile attributes that you can configure.
DROP_PROFILE Procedure	This procedure drops an existing AI profile.
ENABLE_PROFILE Procedure	This procedure enables an AI profile to use in the current database.
DISABLE_PROFILE Procedure	This procedure disables an AI profile in the current database.
SET_ATTRIBUTE Procedure	This procedure sets AI profile attributes.
SET_PROFILE Procedure	This procedure sets AI profile for the current database.
GENERATE Function	This function generates a SQL statement using AI to translate.
GENERATE_SYNTHETIC_DATA Function	This function generates synthetic data.
ENABLE_DATA_ACCESS Procedure	Use this procedure to enable sending data to your LLM.
DISABLE_DATA_ACCESS Procedure	Use this procedure to disable sending data to your LLM.
CREATE_VECTOR_INDEX Procedure	This procedure creates a vector index in the specified vector database, and populates it with data from an object store using an asynchronous scheduler job.
DROP_VECTOR_INDEX Procedure	This procedure removes a vector store index. It normally removes the vector store index object and deletes the vector database.
DISABLE_VECTOR_INDEX Procedure	This procedure disables a vector index object in the current database. When disabled, an AI profile cannot use the vector index, and the system does not load data into the vector store.
ENABLE_VECTOR_INDEX Procedure	This procedure enables or activates a previously disabled vector index object.
UPDATE_VECTOR_INDEX Procedure	This procedure updates an existing vector store index with a specified value of the vector index attribute.
Vector Index Attributes	Provides vector index profile attributes that you can configure.
CREATE_CONVERSATION Procedure	This procedure helps you to create a conversation.
CREATE_CONVERSATION Function	This function helps you to create a conversation and use the conversation ID in other procedures.
CREATE_CONVERSATION Attributes	Use the conversation attributes to customize your conversations.
UPDATE_CONVERSATION Procedure	This procedure updates an existing procedure with user-specified parameters.
SET_CONVERSATION_ID Procedure	This procedure sets conversation support in the current session.
GET_CONVERSATION_ID Function	This procedure helps you to get the conversation_id parameter.

Subprogram	Description
CLEAR_CONVERSATION_ID Procedure	This procedure helps you to clear any conversation_id set in the current session.
DELETE_CONVERSATION_PROMPT Procedure	This procedure deletes a particular prompt.
DROP_CONVERSATION Procedure	This procedure deletes an entire conversation and its metadata.
FEEDBACK Procedure	Use this procedure to potentially improve query generation accuracy by providing a feedback to Select AI.
Vector Index for FEEDBACK	This is a default vector index created when you first use feedback.

CREATE_PROFILE Procedure

The procedure creates a new AI profile for translating natural language prompts to SQL statement.

Syntax

```
DBMS_CLOUD_AI.CREATE_PROFILE
  profile_name      IN  VARCHAR2,
  attributes        IN  CLOB      DEFAULT NULL,
  status            IN  VARCHAR2  DEFAULT NULL,
  description       IN  CLOB      DEFAULT NULL
);
```

Parameters

Parameter	Description
profile_name	A name for the AI profile. The profile name must follow the naming rules of Oracle SQL identifier. Maximum length of profile name is 125 characters. This is a mandatory parameter.
attributes	Profile attributes in JSON format. See AI Profile Attributes for more details. The default value is NULL.
status	Status of the profile. The default value is enable.
description	Description for the AI profile. The default value is NULL.

Example

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name      => 'OpenAI',
    attributes        => JSON_OBJECT('provider' value 'openai',
                                     'credential_name' value
                                     'openai_cred'),
```

```

        description      => 'AI profile to use OpenAI for SQL translation'
    );
END;
/

```

Profile Attributes

Attributes of an AI profile help to manage and configure the behavior of the AI profile. Some attributes are optional and have a default value.

Attributes

Attribute Name	Description
azure_deployment_name	Name of the Azure OpenAI Service deployed model. The name can only include alphanumeric characters, underscore character (_) and a hyphen (-) character. The name cannot end with an underscore (_) or a hyphen (-). To know how to get the azure_deployment_name, see Create and deploy an Azure OpenAI Service resource .
azure_resource_name	Name of the Azure OpenAI Service resource. The resource name can only include alphanumeric characters and hyphens, and can't start or end with a hyphen. To know how to get the azure_resource_name, see Create and deploy an Azure OpenAI Service resource .
comments	Include column comments in the metadata used for translating natural language prompts using AI. BOOLEAN data type is supported. The valid values are TRUE or FALSE for a string with VARCHAR2 data type.
conversation	A VARCHAR2 attribute that indicates if conversation history is enabled for a profile. Only OpenAI and Azure OpenAI Service support conversation history. Valid values are true or false. The default value is false. The values are not case sensitive.
credential_name	The name of the credential to access the AI provider APIs. Credential using bearer tokens can be created by using the provider name as the user name and bearer token as the password. This is a mandatory attribute. See CREATE_CREDENTIAL Procedure.

Note

Boolean values are not applicable in the DBMS_CLOUD_AI.SET_ATTRIBUTE procedure when setting a single attribute because attribute_value parameter is of CLOB datatype.

Attribute Name	Description
embedding_model	The embedding model defined in the AI profile. The following are the supported AI providers for the embedding models with their default values: - OCI GenAI: cohere.embed-english-v3.0 - OpenAI: text-embedding-ada-002 - Azure OpenAI: text-embedding-ada-002 - Cohere: embed-english-v2.0 - Google: text-embedding-004
	 Note The embedding_model parameter is not applicable for Anthropic and Hugging Face.
max_tokens	Denotes the number of tokens to predict per generation. Default is 1024. See Tokens and Tokenizers for more details.
model	The name of the AI model being used to generate responses. Supported models for: OpenAI: gpt-4, gpt-4-0613, gpt-4-32k, gpt-4-32k-0613, gpt-3.5-turbo (default), gpt-3.5-turbo-0613, gpt-3.5-turbo-16k, gpt-3.5-turbo-16k-0613 Cohere: command (default), command-nightly (experimental), command-light, and command-light-nightly (experimental). Smaller, "light" models are faster, while larger models perform better. Custom models can also be supplied with their full ID. OCI Generative AI: For OCI Generative AI: cohere.command (default). Pretrained models for OCI Generative AI are all supported by Select AI. Custom models can also be supplied with their full OCIDs. To know more about supported models in OCI Generative AI, see Pretrained Foundational Models in Generative AI. AWS: Amazon Bedrock foundation models require access permissions through the Amazon Bedrock console. See the AWS documentation to get the modelID.
	 Note This parameter is not used for Azure as the model is determined when you create your deployment in the Azure OpenAI Service portal.

Attribute Name	Description
object_list	Array of JSON objects specifying the owner and object names that are eligible for natural language translation to SQL. To include all objects of a given user, omit the "name" and only specify the "owner" key in the JSON object. For translation natural language to SQL, the object name, object owner, object columns and comments are sent to the AI provider using HTTPS requests. Avoid specifying objects with sensitive object name, column names or comments in the object list. AI providers may have limit on the size of metadata allowed in translation requests. Consider limiting the list of objects suitable for the natural language prompts by your application users. Format: <pre>[{"owner": "SH", "name": "SALES", {"owner": "TEST_USER"}]</pre>
oci_compartment_id	Specifies the OCID of the compartment you are permitted to access when calling the OCI Generative AI service. The compartment ID can contain alphanumeric characters, hyphens and dots. The default is the compartment ID of the Autonomous AI Database.
oci_endpoint_id	This attributes indicates the endpoint OCID of the Oracle dedicated AI hosting cluster. The endpoint ID can contain alphanumeric characters, hyphens and dots. To find the endpoint OCID, see Getting an Endpoint's Details in Generative AI. When you want to use the Oracle dedicated AI cluster, you must provide the endpoint OCID of the hosting cluster. By default, the endpoint ID is empty and the model is on-demand on a shared infrastructure.
oci_runtime_type	This attribute indicates the runtime type of the provided model. This attribute is required when the model attribute is specified and is applicable to OCI Generation Models. All permitted values can be found in OCI Generative AI runtimeType. See LlmInferenceRequest Reference. The supported values are: • COHERE • LLAMA
provider	AI provider for the AI profile. Supported providers: • openai • cohere • azure • oci • google • anthropic • huggingface • aws This is a mandatory attribute.

Attribute Name	Description
region	This attribute indicates the location of the Generative AI cluster that you want to use. The region can contain alphanumeric characters and hyphen characters.
	Note The Oracle Generative AI cluster is available in Chicago, Frankfurt, and London regions. See Pretrained Foundational Models in Generative AI.
	The default region is us-chicago-1. The default region for AWS is us-east-1.
stop_tokens	The generated text will be terminated at the beginning of the earliest stop sequence. Sequence will be incorporated into the text. The attribute value must be a valid array of string values in JSON format. stop_tokens takes a JSON array as input. To learn more about stop tokens or stop sequences, see OpenAI or Cohere documentation.
temperature	Sampling from Generate Text models incorporates randomness, so that the same prompt may yield different outputs each time you hit "generate". Temperature is a non-negative float number used to tune the degree of randomness. Lower temperatures mean less random generations. See Temperature for more details. This parameter is applicable to all the supported service providers.

The following example is using Cohere as the provider and displays custom profile attributes:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name => 'COHERE',
    attributes =>
      '{"provider": "cohere",
        "credential_name": "COHERE_CRED",
        "object_list": [{"owner": "ADB_USER"}],
        "max_tokens": 512,
        "stop_tokens": [";"],
        "model": "command-nightly",
        "temperature": 0.5,
        "comments": true
      }');
END;
/
```

The following example shows custom profile attributes using OCI Generative AI:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
    profile_name =>
      'GENAI',
    attributes => '{"provider":
      "oci",
        "credential_name": "GENAI_CRED",
```

```
        "object_list": [{"owner": "SH", "name": "customers"},
                        {"owner": "SH", "name": "countries"},
                        {"owner": "SH", "name": "supplementary_demographics"},
                        {"owner": "SH", "name": "profits"},
                        {"owner": "SH", "name": "promotions"},
                        {"owner": "SH", "name": "products"}],
        "oci_compartment_id": "ocid1.compartment.oc1...",
        "oci_endpoint_id": "ocid1.generativeaiendpoint.oc1.us-chicago-1...",
        "region": "us-chicago-1",
        "model": "cohere.command-light",
        "oci_runtime_type": "COHERE"
    }');
END;
/
```

DROP_PROFILE Procedure

The procedure drops an existing AI profile. If the profile does not exist, then the procedure throws an error.

Syntax

```
DBMS_CLOUD_AI.DROP_PROFILE(
    profile_name      IN    VARCHAR2,
    force             IN    BOOLEAN DEFAULT FALSE
);
```

Parameters

Parameter	Description
profile_name	Name of the AI profile
force	If TRUE, then the procedure ignores errors if AI profile does not exist. The default value for this parameter is FALSE.

Example

```
BEGIN
    DBMS_CLOUD_AI.DROP_PROFILE(profile_name => 'OPENAI');
END;
/
```

Usage Notes

Use force to drop a profile and ignore errors if AI profile does not exist.

ENABLE_PROFILE Procedure

This procedure enables the AI profile that the user specifies. The procedure changes the status of the AI profile to ENABLED.

Syntax

```
DBMS_CLOUD_AI.ENABLE_PROFILE(  
    profile_name          IN   VARCHAR2  
);
```

Parameters

Parameter	Description
profile_name	Name for the AI profile to enable This parameter is mandatory.

Example to Enable AI Profile

```
BEGIN  
    DBMS_CLOUD_AI.ENABLE_PROFILE(  
        profile_name      => 'OPENAI'  
    );  
END;  
/
```

DISABLE_PROFILE Procedure

This procedure disables the AI profile in the current database. The status of the AI profile is changed to DISABLED by this procedure.

Syntax

```
DBMS_CLOUD_AI.DISABLE_PROFILE(  
    profile_name IN VARCHAR2  
);
```

Parameters

Parameter	Description
profile_name	Name for the AI profile. This parameter is mandatory.

Example

```
BEGIN  
    DBMS_CLOUD_AI.DISABLE_PROFILE(  
        profile_name      => 'OPENAI'  
    );
```

```
END;  
/
```

SET_ATTRIBUTE Procedure

This procedure enables you to set AI profile attributes.

Syntax

```
DBMS_CLOUD_AI.SET_ATTRIBUTE(  
    profile_name      IN   VARCHAR2,  
    attribute_name    IN   VARCHAR2,  
    attribute_value    IN   CLOB  
);
```

Parameters

Only the owner can set or modify the attributes of the AI profile. For a list of supported attributes, see [Profile Attributes](#).

Parameter	Description
profile_name	Name of the AI profile for which you want to set the attributes. This parameter is mandatory.
attribute_name	Name of the AI profile attribute This parameter is mandatory.
attribute_value	Value of the profile attribute. The default value is NULL.

Example

```
BEGIN  
  DBMS_CLOUD_AI.SET_ATTRIBUTE(  
    profile_name => 'OPENAI',  
    attribute_name => 'credential_name',  
    attribute_value => 'OPENAI_CRED_NEW'  
  );  
END;  
/
```

SET_PROFILE Procedure

This procedure sets AI profile for current session.

After setting an AI profile for the database session, any SQL statement with the prefix SELECT AI is considered a natural language prompt. Depending on the action specified with the AI prefix, a response is generated using AI. To use the AI prefix, see [Use AI Keyword to Enter Prompts](#). Optionally, it is possible to override the profile attributes or modify attributes by specifying them in JSON format. See [SET_ATTRIBUTE Procedure](#) for setting the attributes.

The AI profile can only be set for current session if the owner of the AI profile is the session user.

To set an AI profile for all sessions of a specific database user or all user sessions in the database, consider using a database event trigger for AFTER LOGON event on the specific user or the entire database. See *CREATE TRIGGER Statement* in Oracle Database 19c Database PL/SQL Language Reference or [Oracle Database 26ai Database PL/SQL Language Reference](#) for more details.

Syntax

```
DBMS_CLOUD_AI.SET_PROFILE(  
    profile_name      IN  VARCHAR2,  
);
```

Parameters

Parameter	Description
profile_name	A name for the AI profile in the current session. This parameter is mandatory.

Example

```
BEGIN  
    DBMS_CLOUD_AI.SET_PROFILE(  
        profile_name => 'OPENAI'  
    );  
END;  
/
```

GENERATE Function

This function provides AI translation when using a stateless database connection. With your existing AI profile, you can use this function to perform the supported actions such as showsql, runsql, explainsql, narrate, summarize, and chat. The default action is showsql. Overriding some or all of the profile attributes is also possible using this function.

Syntax

```
DBMS_CLOUD_AI.GENERATE(  
    prompt           IN  CLOB,  
    profile_name     IN  VARCHAR2 DEFAULT NULL,  
    action           IN  VARCHAR2 DEFAULT NULL,  
    attributes       IN  CLOB      DEFAULT NULL,  
    params           IN  CLOB  
    ) RETURN CLOB;
```

Parameters

Parameter	Description
prompt	Natural language prompt to translate using AI. The prompt can include <code>SELECT AI <action></code> as the prefix. The action can also be supplied separately as an "action" parameter. The <i>action</i> supplied in the prompt overrides the <code>action</code> parameter. Default action is <code>showsql</code>. This parameter is mandatory.

Parameter	Description
profile_name	Name of the AI profile. This parameter is optional if an AI profile is already set in the session using <code>DBMS_CLOUD_AI.SET_PROFILE</code>. The default value is <code>NULL</code>. The following conditions apply: • If a profile is set in the current session, the user may omit <code>profile_name</code> argument in the <code>DBMS_CLOUD_AI.GENERATE</code> function. • If the <code>profile_name</code> argument is supplied in the <code>DBMS_CLOUD_AI.GENERATE</code> function, it overrides any value set in the session using the <code>DBMS_CLOUD_AI.SET_PROFILE</code> procedure. • If there is no profile set in the session using the <code>DBMS_CLOUD_AI.SET_PROFILE</code> procedure, the <code>profile_name</code> argument must be supplied in the <code>DBMS_CLOUD_AI.GENERATE</code> function.

Note

For Database Actions, you can either specify `profile_name` argument in `DBMS_CLOUD_AI.GENERATE` or you can run two steps as a PL/SQL script: `DBMS_CLOUD_AI.SET_PROFILE` and `DBMS_CLOUD_AI.GENERATE`.

```
EXEC
DBMS_CLOUD_AI.set_profile('OPENAI');
```

```
-----
-----
```

```
SELECT
DBMS_CLOUD_AI.GENERATE(prompt
=> 'how many customers',
```

```
profile_name => 'OPENAI',
```

```
action      => 'runsql')
FROM dual;
```

```
-----
-----
```

```
SELECT
DBMS_CLOUD_AI.GENERATE(prompt
=> 'how many customers',
```

```
profile_name => 'OPENAI',
```

```
action      => 'showsql')
FROM dual;
```

```
-----
-----
```

```
SELECT
DBMS_CLOUD_AI.GENERATE(prompt
=> 'how many customers',
```


Parameter	Description
	<pre>profile_name => 'OPENAI', action => 'explainsql') FROM dual; ----- ----- SELECT DBMS_CLOUD_AI.GENERATE(prompt => 'how many customers', profile_name => 'OPENAI', action => 'narrate') FROM dual; ----- ----- SELECT DBMS_CLOUD_AI.GENERATE(prompt => 'what is oracle autonomous database', profile_name => 'OPENAI', action => 'chat') FROM dual;</pre> See Executing SQL Statements in the Code Editor for more information.
action	Action for translating natural prompt using AI. The supported actions include: - runsql(default) - showsql - explainsql - narrate - summarize - translate - chat Descriptions of actions are included in Use AI Keywords to Enter Prompts.
attributes	Override specific AI profile attributes by supplying attributes in JSON format. See Profile Attributes for more details.
params	Specify conversation parameter. See CREATE CONVERSATION Attributes. You can specify only the following parameter: conversation_id: The default value is the conversation id set in the current session. This is not a mandatory parameter.

Examples

Example: Using the GENERATE Function for Select AI Actions

The following examples illustrate `runsql`, `showsql`, `explainsql`, `narrate`, `summarize`, `translate`, and `chat` actions that can be used with the `DBMS_CLOUD_AI.GENERATE` function.

See also [Use AI Keywords to Enter Prompts](#) for more details.

An example with `runsql` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt      => 'how many customers',
                             profile_name => 'OPENAI',
                             action       => 'runsql')
FROM dual;
```

An example with `showsql` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt      => 'how many customers',
                             profile_name => 'OPENAI',
                             action       => 'showsql')
FROM dual;
```

An example with `explainsql` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt      => 'how many customers',
                             profile_name => 'OPENAI',
                             action       => 'explainsql')
FROM dual;
```

An example with `narrate` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt      => 'how many customers',
                             profile_name => 'OPENAI',
                             action       => 'narrate')
FROM dual;
```

An example with `chat` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt      => 'what is oracle autonomous
database',
                             profile_name => 'OPENAI',
                             action       => 'chat')
FROM dual;
```

`summarize`

```
SELECT DBMS_CLOUD_AI.GENERATE(
    prompt => TO_CLOB(
        DBMS_CLOUD.GET_OBJECT(
            credential_name => 'STORE_CRED',
            object_uri => 'https://objectstorage.ca-
toronto-1.oraclecloud.com' ||
                                '/n/namespace-string/b/bucketname/o/
data_folder/' ||
                                'summary/test_4000_words.txt')),
    profile_name => 'GENAI_LLAMA',
```

```
        action => 'SUMMARIZE')
from DUAL;
```

Result:

The music streaming industry, led by Spotify, has revolutionized the way people consume music, with streaming accounting for 80% of the American recording industry's revenue. However, this shift has also complicated the lives of artists trying to survive in an on-demand, hyper-abundant present. Spotify's business model, which pays royalties based on an artist's popularity, has led to concerns about the fairness of the system, with some artists earning little to no royalties. The company's dominance has also changed the way people listen to music, with a focus on convenience and personalized playlists. Liz Pelly's book, "Mood Machine: The Rise of Spotify and the Costs of the Perfect Playlist," explores the impact of Spotify's rise on the music industry and listeners, arguing that the platform's emphasis on affect and mood has led to a homogenization of music and a loss of autonomy for listeners. As the music industry continues to evolve, questions remain about the future of music creation and consumption, and whether artists will be able to thrive in a system that prioritizes convenience and profit over artistic expression.

These examples show translate action:

The following examples shows using translate action in the prompt.

Note

Your AI profile must specify target language.

```
SELECT DBMS_CLOUD_AI.GENERATE('select ai translate text to be translated')
       FROM dual;
```

The following example shows translate action supplied in the `DBMS_CLOUD_AI.GENERATE` function along with `target_language` and `source_language`. This example uses generative AI translation. The input text this is a document in English (`source_language: "en"`) is translated into French (`target_language: "fr"`).

```
DECLARE
    l_attributes clob := '{"target_language": "fr", "source_language":
"en"}';
    output clob;
BEGIN
    output := DBMS_CLOUD_AI.GENERATE(
        prompt          => 'this is a document',
        profile_name     => 'oci_translate',
        action          => 'translate',
        attributes       => l_attributes
    );
```

Using the GENERATE Function in a Procedure

You can use `DBMS_CLOUD_AI.GENERATE` in a procedure and run the function. The following example takes an `ai_prompt`, `profile_name`, and `action` as input parameters and calls `DBMS_CLOUD_AI.GENERATE`

```
create or replace FUNCTION call_select_ai (ai_prompt  IN VARCHAR2,
                                          ai_profile IN VARCHAR2,
                                          ai_action  IN VARCHAR2) -- valid
for 'chat', 'narrate', 'showsql'
RETURN CLOB AS sai_resp clob;

BEGIN
  sai_resp := DBMS_CLOUD_AI.GENERATE(prompt      => ai_prompt,
                                     profile_name => ai_profile,
                                     action       => ai_action);

  return(sai_resp);
END call_select_ai;
```

GENERATE_SYNTHETIC_DATA Function

Use this procedure to generate synthetic data for a single table, multiple tables or a full schema.

The following is the syntax to generate synthetic data for a single table.

Syntax

```
DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
  profile_name      IN  VARCHAR2,
  object_name       IN  DBMS_ID,
  owner_name        IN  DBMS_ID,
  record_count      IN  NUMBER,
  user_prompt       IN  CLOB DEFAULT NULL,
  params            IN  CLOB DEFAULT NULL
);
```

The following is the syntax to generate synthetic data for multiple tables.

```
DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
  profile_name      IN  VARCHAR2,
  object_list       IN  CLOB,
  params            IN  CLOB DEFAULT NULL
);
```

If you do not want table data or vector search documents to be sent to an LLM, a user with administrator privileges can disable such access for all users of the given database. This, in effect, disables the `narrate` action.

Parameters

Parameter	Description
profile_name	The AI profile containing necessary LLM service information. This can be created by CREATE_PROFILE Procedure .
object_name	Specify a table name to populate synthetic data. - SELECT and INSERT privilege on the table objects are needed for the user using it. - The table is either empty or have records in it.
owner_name	Specify the database user who owns the referenced object. If no specific owner is provided, the procedure defaults to using the schema of the user running it.
record_count	The number of records to be synthetically generated.
user_prompt	Additional information that a user can mention to generate synthetic data. For example, to generate a record for a table called MOVIE with a release_date column, the user_prompt can be: <i>the release date for the movies should be in 2019</i>
params	Optional attributes provided in JSON object string format to modify the behavior of an API. See Optional Parameters .
object_list	Use this parameter for generating synthetic data on multiple tables. This parameter takes in table object information along with its arguments and contains the same arguments provided in the single table. See object_list Parameters .

Optional Parameters

Parameter	Value Datatype	Value	Description
sample_rows	Number	0 <= sample_rows <= 100	Specify the number of rows from the table to use as a sample to guide the LLM in data generation. A value of 0 means no sample rows will be used. The default value is 0.
table_statistics	Boolean	- True - False	Enable or disable the use of table statistics information. The default value is False.

Parameter	Value Datatype	Value	Description
priority	String	Valid values: • HIGH • MEDIUM • LOW	Assign a priority value that defines the number of parallel requests sent to the LLM for generating synthetic data. Tasks with a higher priority will consume more database resources and complete faster. The default value is HIGH • HIGH: Specifies the number of parallel LLM requests based on the database's ECPU count (or OCPU count if your database uses OCPUs). • MEDIUM: Sets the number of concurrent processes according to the concurrency limit for Medium service. The default value is 4. • LOW: Runs the pipeline job in a serial order, without parallel processing. The maximum number of concurrent parallel processes used for synthetic data generation is limited to 64.
comments	Boolean	• True • False	Enable or disable sending comments to the LLM to guide data generation. The default value is False.

object_list Parameters

Parameter	Value Datatype	Mandatory	Description
owner	String	Yes	Specifies the database user who owns the object being referenced. If no specific owner is provided, the procedure will default to using the schema of the user running it.
name	String	No	Specify a table name to populate synthetic data. SELECT and INSERT privilege on the table objects are needed for the user using it. The table is either empty or have records in it.
record_count	Number	No	The number of records to be synthetically generated. Provide a number greater than 0. Supply record_count or record_count_percentage.

Parameter	Value Datatype	Mandatory	Description
record_count_percentage	Number	No	The percentage of number of records to be synthetically generated. Provide a number greater than 0. For a Metadata Clone database, where the table metadata including statistics is preserved, the record_count_percentage parameter is supported. Supply record_count or record_count_percentage. When using the record_count_percentage parameter, the final record count in the table is calculated as: $\text{Original\_Num\_Rows} \times \text{record\_count\_percentage}$
user_prompt	String	No	Same as user_prompt in Parameters . The user_prompt is associated with a specific table object.

Examples

The following examples show the DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA function for generating synthetic data for a single table and multiple tables. For a complete example and to view more examples, see [#unique_1027](#).

```

BEGIN
    DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
        profile_name => 'GENAI',
        object_name  => 'Director',
        owner_name   => 'ADB_USER',
        record_count => 5
    );
END;
```



```
/
PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_AI.GENERATE_SYNTHETIC_DATA(
    profile_name => 'GENAI',
    object_list => '[{"owner": "ADB_USER", "name":
"Director", "record_count":5},
    {"owner": "ADB_USER", "name":
"Movie_Actor", "record_count":5},
    {"owner": "ADB_USER", "name":
"Actor", "record_count":10},
    {"owner": "ADB_USER", "name":
"Movie", "record_count":5, "user_prompt": "all movies are released in 2009"}]'
  );
END;
/
PL/SQL procedure successfully completed.
```

ENABLE_DATA_ACCESS Procedure

This procedure enables sending data to LLM for applicable Select AI features, which is the default behavior. Only an administrator can run this procedure.

This procedure controls data access for the following Select AI capabilities:

- narrate action
- Retrieval Augmented Generation (RAG)
- Synthetic Data Generation

Syntax

```
DBMS_CLOUD_AI.ENABLE_DATA_ACCESS();
```

Parameters

This procedure does not require any parameters.

Example to Enable Data Access

```
BEGIN
  DBMS_CLOUD_AI.ENABLE_DATA_ACCESS();
END;
/
```

DISABLE_DATA_ACCESS Procedure

This procedure disables sending data to LLM for applicable Select AI features. Only an administrator can run this procedure.

This procedure limits the following Select AI capabilities:

- narrate action

- Retrieval Augmented Generation (RAG)
- Synthetic Data Generation

Syntax

```
DBMS_CLOUD_AI.DISABLE_DATA_ACCESS();
```

Parameters

This procedure does not require any parameters.

Example to Disable Data Access

```
BEGIN
  DBMS_CLOUD_AI.DISABLE_DATA_ACCESS();
END;
/
```

CREATE_VECTOR_INDEX Procedure

This procedure creates a vector index in the specified vector database, and populates it with data from an object store using an asynchronous scheduler job.

Syntax

```
PROCEDURE CREATE_VECTOR_INDEX(
  index_name      IN VARCHAR2,
  attributes      IN CLOB      DEFAULT NULL,
  status         IN VARCHAR2  DEFAULT NULL,
  description     IN CLOB      DEFAULT NULL
);
```

Parameters

Parameter	Description
index_name	Name of the vector index. The vector index name must follow the naming rules of Oracle SQL identifier. Maximum length of vector store name is 125 characters. This is a mandatory parameter.
attributes	Custom attributes for the vector index in JSON. To see a list of configurable parameters, see Vector Index Attributes . The default value is NULL.
status	Status of the vector index. The possible values are: • Enabled • Disabled The default value is Disabled.
description	Description for the vector index. The default value is NULL.

Example

The following example demonstrates how to create a vector index and configure the attributes as JSON parameters.

```
BEGIN
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name      => 'MY_INDEX'
        attributes      => JSON_OBJECT(
            'vector_db_provider' value 'oracle',
            'vector_table_name' value
'oracle_mycollection',
            'profile_name'      value 'OCIGENAI',
            'location'          value
            'https://objectstorage.us-phoenix-1.' ||
            'oraclecloud.com/n/mynamespace/b/mybucket',
            'object_store_credential_name' value
'OS_CRED',
            'chunk_size'        value 2048,
            'chunk_overlap'     value 256,
            'refresh_rate'      value 720)
        );
END;
/
```

DROP_VECTOR_INDEX Procedure

This procedure removes a vector store index. It normally removes the vector store index object and deletes the vector store. If set to FALSE, the argument `include_data` ensures the procedure only removes the vector store index object while retaining the vector store.

Syntax

```
PROCEDURE DROP_VECTOR_INDEX(
    index_name      IN  VARCHAR2,
    include_data    IN  BOOLEAN DEFAULT TRUE,
    force           IN  BOOLEAN DEFAULT FALSE
);
```

Parameters

Parameter	Description
<code>index_name</code>	Name of the vector index. The vector index name must follow the naming rules of Oracle SQL identifier. Maximum length of vector store name is 125 characters. This is a mandatory parameter.
<code>include_data</code>	Indicates whether to delete both the customer's vector store and vector index along with the vector index object. Possible values: - TRUE - FALSE The default value is TRUE.

Parameter	Description
force	Indicates whether to ignore errors that occur if the vector index does not exist. Possible values: • TRUE • FALSE If set to TRUE, this parameter bypasses any errors that occur if the vector index does not exist. The default value is FALSE.

Example

```
BEGIN
DBMS_CLOUD_AI.DROP_VECTOR_INDEX(
    index_name => 'MY_INDEX',
    include_data => FALSE,
    force      => TRUE
);
END;
/
```

DISABLE_VECTOR_INDEX Procedure

This procedure disables a vector index object in the current database. When disabled, an AI profile cannot use the vector index, and the system does not load data into the vector store as new data is added to the object store and does not perform indexing, searching or querying based on the index.

Syntax

```
DBMS_CLOUD_AI.DISABLE_VECTOR_INDEX(
    index_name      IN  VARCHAR2
);
```

Parameters

Parameter	Description
index_name	Name of the vector index. The vector index name must follow the naming rules of Oracle SQL identifier. Maximum length of vector store name is 125 characters. This is a mandatory parameter.

Example

You can disable a vector index by providing the name of the vector index.

```
BEGIN
    DBMS_CLOUD_AI.DISABLE_VECTOR_INDEX(index_name => 'MY_INDEX');
```

```
END;  
/
```

ENABLE_VECTOR_INDEX Procedure

This procedure enables or activates a previously disabled vector index object. Generally, when you create a vector index, by default it is enabled such that the AI profile can use it to perform indexing and searching.

When enabled, a vector index allows an AI profile to use it for loading new data from an object store into a vector store at a user-specified refresh rate. You can specify the `refresh_rate` parameter through the JSON object list. To configure the JSON attributes, see [Vector Index Attributes](#).

Syntax

```
DBMS_CLOUD_AI.ENABLE_VECTOR_INDEX(  
    index_name      IN  VARCHAR2  
);
```

Parameters

Parameter	Description
<code>index_name</code>	Name of the vector index. The vector index name must follow the naming rules of Oracle SQL identifier. Maximum length of vector store name is 125 characters. This is a mandatory parameter.

Example

You can enable or activate a vector index by specifying the vector index name as follows:

```
BEGIN  
    DBMS_CLOUD_AI.ENABLE_VECTOR_INDEX(index_name => 'MY_INDEX');  
END;  
/
```

UPDATE_VECTOR_INDEX Procedure

This procedure updates an existing vector store index with a specified value of the vector index attribute.

It is overloaded to accept:

- attribute values of various types.
- vector index attributes as a JSON document and updates one or more attributes of an existing vector store index with the specified attribute name and value pair.

Syntax

```
DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX(  
    index_name      IN  VARCHAR2,
```

```

        attributes          IN  CLOB
    );

    DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX(
        index_name          IN  VARCHAR2,
        attribute_name       IN  VARCHAR2,
        attribute_value      IN  VARCHAR2
    );

    DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX(

        index_name          IN  VARCHAR2,
        attribute_name       IN  VARCHAR2,
        attribute_value      IN  CLOB          DEFAULT NULL
    );

```

Parameters

Parameter	Description
index_name	Name of the vector index. The vector index name must follow the naming rules of Oracle SQL identifier. Maximum length of vector store name is 125 characters. This is a mandatory parameter.
attributes	Specifies vector index attributes in JSON format. This is a mandatory parameter.
attribute_name	Name of the custom attributes specified as JSON parameters in CREATE_VECTOR_INDEX procedure. You cannot modify the following attributes: • location • vector_table_name • chunk_size • chunk_overlap • vector_distance_metric • vector_dimension This is a mandatory parameter.
attribute_value	User specified value for the custom attribute_name. Value can be CLOB, NUMBER or VARCHAR2 type. The default value is NULL.

Note

Use either the attributes parameter to specify attribute_name and value pairs in JSON format or the attribute_name and attribute_value parameters together.

Examples

```

BEGIN
    DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX(
        index_name          => 'MY_INDEX',
        attribute_name       => 'object_storage_credential_name',
        attribute_value      => 'NEW_CRED'
    );

```

```
);  
END;  
/
```

The following example accepts NUMBER type as the attribute_value.

```
BEGIN  
  DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX(  
    index_name      => 'MY_INDEX',  
    attribute_name   => 'match_limit',  
    attribute_value  => 10  
  );  
END;  
/
```

The following example accepts VARCHAR2 type as the attribute_value.

```
BEGIN  
  DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX(  
    index_name      => 'MY_INDEX',  
    attribute_name   => 'profile_name',  
    attribute_value  => 'AI_PROF2'  
  );  
END;  
/
```

The following example accepts attributes in JSON format.

```
BEGIN  
  DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX(  
    index_name => 'MY_VECTOR_INDEX',  
    attributes => '{"match_limit": 10,  
                  "refresh_rate": 30}'  
  );  
END;  
/
```

Vector Index Attributes

Attributes of a vector index help to manage and configure the behavior of the vector index. You can add custom index attributes as necessary. Some attributes are optional and have a default value.

Attributes

Attribute Name	Value	Description
chunk_size	1024 (default)	Text size of chunking the input data. For text data, this means the number of characters.
chunk_overlap	128 (default)	Specifies the amount of overlapping characters between adjacent chunks of text. This attribute is useful for ensuring contextual continuity and accuracy in text processing by allowing overlaps between segments, which helps prevent loss of contextual information at chunk boundaries.

Attribute Name	Value	Description
location	NA	This parameter specifies source file URI or directories and source files. Wildcard patterns are supported for both source file URIs and directories. Cloud source file URIs: You can specify a source file URI for bucket or subfolder. You can use wildcards to specify subfolders or file names. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. Example using wild cards: <pre>location_uri => 'https:// objectstorage.my\$region.oraclecloud .com/n/namespace-string/b/ bucketname/o/year=????/month=??/ *.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. Directory: You can specify one directory and file name. The format to specify a directory is: MY_DIR:filename.ext. By default the directory name MY_DIR is a database object and is case-insensitive. The file name is case sensitive. You can only use wildcards to specify file names in a directory. The character * can be used as the wildcard for multiple characters, and the character ? can be used as the wildcard for a single character. For example: MY_DIR: * or MY_DIR:test?. Use double quotes to specify a case-sensitive directory name. For example: "my_dir1":*, "my_dir2":Test? To include a quote character, use two quotes. For example: MY_DIR: ' 'filename.ext. This specifies the filename starts with a quote ('). The files in this location can be documents in formats such as PDF, DOC, JSON, XML, or HTML. See Supported Document Formats.
match_limit	5 (default)	Specifies the maximum number of results to return in a vector search query, controlling the output size and improving the efficiency of data retrieval operations.
object_storage_credential_name	NA	Specifies the name of the credentials for accessing an object storage.

Attribute Name	Value	Description
pipeline_name	<vector_index_name>\$VEPIPELINE	Specifies the name of the vector index data load pipeline. This attribute is automatically set for the vector index, you cannot specify or modify. The pipeline name can be used to monitor the vector index data load using Monitor and Troubleshoot Pipelines.
profile_name	NA	Name of the AI profile which is used for embedding resource data and user prompts.
refresh_rate	1440 minutes (default)	Interval of updating data in the vector store. The unit is minutes.
similarity_threshold	0 (default)	Defines the minimum level of similarity required for two items to be considered a match, useful for filtering results in matching algorithms to ensure relevance.
vector_distance_metric	A string corresponding to one of the values specified in the description.	Specifies the type of distance calculation used to compare vectors in a database, determining how similarity between items is quantified. Valid values for Oracle 23ai: • EUCLIDEAN • L2_SQUARED (EUCLIDEAN_SQUARED) • COSINE (default) • DOT • MANHATTAN • HAMMING
vector_db_provider	oracle	Specifies the provider name that manages and serves as the vector store.
vector_dimension	NA	Specifies the number of elements in each vector within the vector store, defining the size and structure of the data representation.
vector_table_name	<vector_index_name>\$VECTAB (default)	Specifies the name of the table or collection to store vector embeddings and chunked data.

Example: Specify Object Storage URI Location

The following example demonstrates creating a vector index with OCI Generative AI vector store.

```
BEGIN
  DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
    index_name => 'MY_INDEX',
    attributes => '{"vector_db_provider": "oracle",
                  "location": "https://swiftobjectstorage.us-
phoenix-1.oraclecloud.com/v1/my_namespace/my_bucket/my_data_folder",
```

```

        "object_storage_credential_name": "OCI_CRED",
        "profile_name": "OPENAI_ORACLE",
        "vector_dimension": 1024,
        "vector_distance_metric": "cosine",
        "chunk_overlap": 128,
        "chunk_size": 1024
    });
END;
/
/

```

Example: Specify Object Storage URI Location with Wild Card Pattern

This example specifies a wild card pattern (*) in the Object Storage URI as the location parameter. It loads all the CSV files from the Object Storage URI.

```

BEGIN
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name      => 'MY_INDEX',
        attributes      => JSON_OBJECT(
            'vector_db_provider' value 'oracle',
            'vector_table_name' value
'oracle_mycollection',
            'profile_name'      value 'OCIGENAI',
            'location'          value 'https://
objectstorage.myregion.oraclecloud.com/n/my$namespace/b/bucketname/o/
year=????/month=??/file*.csv)',
            'object_storage_credential_name' value
'OS_CRED',
            'chunk_size'       value 2048,
            'chunk_overlap'    value 256,
            'refresh_rate'     value 720)
        );
END;
/

```

Example: Specify Directory Object Location with Wild Card Pattern

This example specifies directory objects in the location parameter using a wild card pattern. It loads all CSV files in the MY_DIR directory.

```

BEGIN
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name      => 'MY_INDEX',
        attributes      => JSON_OBJECT(
            'vector_db_provider' value 'oracle',
            'vector_table_name' value
'oracle_mycollection',
            'profile_name'      value 'OCIGENAI',
            'location'          value 'MY_DIR:*.csv',
            'object_storage_credential_name' value
'OS_CRED',
            'chunk_size'       value 2048,
            'chunk_overlap'    value 256,
            'refresh_rate'     value 720)
        );

```

```
END;
/
```

Example: Specify Case-Sensitive Directory Object Location with Wild Card Pattern

This example specifies a case-sensitive directory objects in the location parameter using a wild card pattern. It loads all CSV files in the My_Dir directory.

```
BEGIN
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name      => 'MY_INDEX',
        attributes      => JSON_OBJECT(
            'vector_db_provider' value 'oracle',
            'vector_table_name' value
'oracle_mycollection',
            'profile_name'      value 'OCIGENAI',
            'location'          value '"My_Dir":*.csv',
            'object_storage_credential_name' value
'OS_CRED',
            'chunk_size'        value 2048,
            'chunk_overlap'     value 256,
            'refresh_rate'      value 720)
        );
END;
/
```

Example: Specify Case-Sensitive Directory Object with all Files as Wild Card Pattern

This example specifies a case-sensitive directory object in the location parameter using a wildcard pattern (*). It loads all files located in the My_Dir directory.

```
BEGIN
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name      => 'MY_INDEX',
        attributes      => JSON_OBJECT(
            'vector_db_provider' value 'oracle',
            'vector_table_name' value
'oracle_mycollection',
            'profile_name'      value 'OCIGENAI',
            'location'          value '"My_Dir":*',
            'object_storage_credential_name' value
'OS_CRED',
            'chunk_size'        value 2048,
            'chunk_overlap'     value 256,
            'refresh_rate'      value 720)
        );
END;
/
```

Example: Specify a File Name Match in the Directory Object

This example specifies a directory object and uses a file name prefix, such as *test*, in the location parameter. It loads all files in the MY_DIR directory whose names begin with *test*.

```
BEGIN
    DBMS_CLOUD_AI.CREATE_VECTOR_INDEX(
        index_name      => 'MY_INDEX',
```

```
attributes => JSON_OBJECT(
    'vector_db_provider' value 'oracle',
    'vector_table_name' value
'oracle_mycollection',
    'profile_name' value 'OCIGENAI',
    'location' value 'MY_DIR:test*',
    'object_storage_credential_name' value
'OS_CRED',
    'chunk_size' value 2048,
    'chunk_overlap' value 256,
    'refresh_rate' value 720)
);
END;
/
```

CREATE_CONVERSATION Procedure

This procedure enables you to create a conversation and automatically set the conversation_id within the procedure.

Note

If you are using DBMS_CLOUD_AI.CREATE_CONVERSATION procedure, you can skip setting the conversation_id as the procedure automatically sets it.

Syntax

```
DBMS_CLOUD_AI.CREATE_CONVERSATION(
    attributes          IN CLOB DEFAULT NULL
);
```

Parameters

Parameter	Description
attributes	Attributes for conversation in JSON format. See CREATE_CONVERSATION Attributes for more details. The default value is NULL.

Example

Example: Create Conversation

The following example shows creating a conversation without any customization.

```
EXEC DBMS_CLOUD_AI.CREATE_CONVERSATION;
```

Result:

PL/SQL procedure successfully completed.

Example: Create Conversation with Custom Attributes

The following example shows creating a conversation with custom parameters such as title, description, retention_days and conversation_length.

```
-- Create conversation with custom attributes
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION(
    attributes => '{"title":"Conversation 1",
                  "description":"this is a description",
                  "retention_days":5,
                  "conversation_length":5}')
    AS conversation_id FROM dual;
```

CREATE_CONVERSATION Function

This function creates a conversation and returns its conversation_id that can be used in other procedures or functions such as DBMS_CLOUD_AI.SET_CONVERSATION_ID and DBMS_CLOUD_AI.GENERATE.

Oracle recommends setting conversation_id to enable conversation. Alternately, you can set conversation_id in the DBMS_CLOUD_AI.GENERATE function.

Note

If you are using DBMS_CLOUD_AI.CREATE_CONVERSATION procedure, you can skip setting the conversation_id as the procedure automatically sets it.

Syntax

```
DBMS_CLOUD_AI.CREATE_CONVERSATION(
    attributes          IN CLOB DEFAULT NULL
) RETURN VARCHAR2;
```

Parameters

Parameter	Description
attributes	Attributes for conversation in JSON format. See CREATE_CONVERSATION Attributes for more details. The default value is NULL.

Example

Example: Create Conversation

The following example shows using DBMS_CLOUD_AI.CREATE_CONVERSATION function to create a conversation without any customization.

```
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION FROM DUAL;
```

Result:

```
CREATE_CONVERSATION
-----
30C9DB6E-EA4D-AFBA-E063-9C6D46644B92
```

Example: Create Conversation with Custom Attributes

The following example shows using DBMS_CLOUD_AI.CREATE_CONVERSATION function to specify attributes such as title, retention_days and conversation_length.

```
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION(
    attributes => '{"title":"This is a test conversation",
                  "retention_days":7,
                  "conversation_length":20}')
FROM DUAL;
```

CREATE_CONVERSATION Attributes

These attributes manage conversation context, including how long to retain it, how many prompts with responses to store or display, and metadata like title and description for reference. Some attributes are optional and have a default value.

Attributes

Attribute Name	Description
	Default Value
title	The user-assigned name for the conversation. If not provided, Select AI will have the LLM generate one when the conversation is first used with a prompt.
	Conversation
description	Provides a user-defined description summarizing the purpose or context of the conversation. If it's not provided, the LLM generates one when the conversation is first used with a prompt and update it again on the 5th use to include more accurate and relevant information.
retention_days	Specify the number of days to retain the conversation history. This is stored in the database from its creation date. If you omit the value, the systems sets it to default value of 7. If you set it to 0, the system retains the conversation until you manually delete it using the DBMS_CLOUD_AI.DROP_CONVERSATION procedure or DBMS_CLOUD.DELETE_ALL_OPERATIONS('CONVERSATION').

Attribute Name	Description
	e f a u l t v a i u e
conversation_length	Specify the number of recent prompts and responses to include with the current prompt. The maximum allowed value is 999. You can override this value by specifying the conversation_length parameter in the DBMS_CLOUD_AI.GENERATE function or by setting it in the AI profile using <code>SELECT AI <ACTION> <PROMPT></code>. Apply the following precedence rules for conversation_length: 1. Value from the attributes argument in <code>DBMS_CLOUD_AI.GENERATE</code> 2. Value set in the conversation takes the next priority 3. Value defined in the AI profile takes the last priority If none of them specify the conversation_length, then the default value is 10.

The following example shows you how you can customize conversation attributes in the `DBMS_CLOUD_AI.CREATE_CONVERSATION` procedure.

```
-- Create conversation with custom attributes
SELECT DBMS_CLOUD_AI.CREATE_CONVERSATION(
    attributes => '{"title":"Conversation 1",
                  "description":"this is a description",
                  "retention_days":5,
                  "conversation_length":5}')
AS conversation_id FROM dual;
```

UPDATE_CONVERSATION Procedure

This procedure updates an existing conversation with a specified value of the conversation attributes.

Syntax

```
DBMS_CLOUD_AI.UPDATE_CONVERSATION(
    conversation_id    IN VARCHAR2,
    attributes         IN CLOB
);
```


Parameters

Parameter	Description
conversation_id	Unique number assigned to a conversation. This is a mandatory parameter.
attributes	Attributes for conversation in JSON format. See CREATE_CONVERSATION Attributes for more details.

Example

```
EXEC DBMS_CLOUD_AI.UPDATE_CONVERSATION(  
  conversation_id => '30C9DB6E-EA4E-AFBA-E063-9C6D46644B92',  
  attributes => '{"retention_days":20,  
                "description":"This a sample description",  
                "title":"Sample title",  
                "conversation_length":20}');
```

Result:

PL/SQL procedure successfully completed.

SET_CONVERSATION_ID Procedure

This procedure sets the current conversation to the specified ID. Subsequent prompts include existing conversation prompts based on the conversation's configured attributes.

Syntax

```
DBMS_CLOUD_AI.SET_CONVERSATION_ID(  
  conversation_id  IN VARCHAR2  
);
```

Parameters

Parameter	Description
conversation_id	Unique number assigned to a conversation in the current session. This parameter is mandatory.

Example

```
EXEC DBMS_CLOUD_AI.SET_CONVERSATION_ID('30C9DB6E-EA4D-AFBA-  
E063-9C6D46644B92');
```

Result:

PL/SQL procedure successfully completed.

GET_CONVERSATION_ID Function

This function returns the conversation ID currently set in the session using either the DBMS_CLOUD_AI.SET_CONVERSATION_ID or DBMS_CLOUD_AI.CREATE_CONVERSATION procedure. If you did not set a conversation, the function returns NULL. If you drop the conversation, the system clears it in the session as well. See [CLEAR_CONVERSATION_ID Procedure](#).

Syntax

```
DBMS_CLOUD_AI.GET_CONVERSATION_ID  
RETURN VARCHAR2;
```

Example

This example displays the conversation ID set in the current session.

```
SELECT DBMS_CLOUD_AI.GET_CONVERSATION_ID;
```

Result:

```
-----  
--  
30C9DB6E-EA4F-AFBA-E063-9C6D46644B92
```

CLEAR_CONVERSATION_ID Procedure

This procedure clears a conversation ID set in the session to disable the conversation feature for SELECT AI <ACTION> <PROMPT>. If you did not set a conversation, the system does not raise any error.

Syntax

```
DBMS_CLOUD_AI.CLEAR_CONVERSATION_ID;
```

Example

This example demonstrates displaying the current conversation ID in the session, clearing the ID, and verifying the change.

```
-- A conversation id is set in the session  
SELECT DBMS_CLOUD_AI.GET_CONVERSATION_ID FROM dual;
```

```
GET_CONVERSATION_ID  
-----  
--  
3A88BFF0-1D7E-B3B8-E063-9C6D46640ECD
```

```
-- Clear the conversation id  
EXEC DBMS_CLOUD_AI.CLEAR_CONVERSATION_ID;
```

PL/SQL procedure successfully completed.

```
-- The conversation id is removed from the session
SELECT DBMS_CLOUD_AI.GET_CONVERSATION_ID FROM dual;
```

```
GET_CONVERSATION_ID
-----
--
```

DELETE_CONVERSATION_PROMPT Procedure

The procedure removes a certain prompt from the conversation.

Syntax

```
DBMS_CLOUD_AI.DELETE_CONVERSATION_PROMPT(
    conversation_prompt_id IN VARCHAR2,
    force                  IN BOOLEAN DEFAULT FALSE
);
```

Parameters

Parameter	Description
conversation_prompt_id	Unique number assigned to a prompt in a conversation. You can find the prompt ID by querying DBA/USER_CLOUD_AI_CONVERSATION_PROMPTS view. See DBMS_CLOUD_AI Views. This is a mandatory parameter.
force	If TRUE, then the procedure ignores errors if the conversation_prompt_id does not exist. The default value for this parameter is FALSE.

Example

```
EXEC DBMS_CLOUD_AI.DELETE_CONVERSATION_PROMPT('30C9DB6E-EA61-AFBA-E063-9C6D46644B92');
```

Result:

PL/SQL procedure successfully completed.

DROP_CONVERSATION Procedure

The procedure removes the conversation and all its associated prompts and the associated responses. Once dropped, the conversation_id becomes invalid. If a conversation is dropped while it's set in the session, it is cleared automatically.

Syntax

```
DBMS_CLOUD_AI.DROP_CONVERSATION(
    conversation_id IN VARCHAR2,
```

```

        force          IN BOOLEAN DEFAULT FALSE
    );

```

Parameters

Parameter	Description
conversation_id	Unique number assigned to a conversation. This is a mandatory parameter.
force	If TRUE, then the procedure ignores errors if the conversation_id does not exist. The default value for this parameter is FALSE.

Example

```
EXEC DBMS_CLOUD_AI.DROP_CONVERSATION('30C9DB6E-EA4D-AFBA-E063-9C6D46644B92');
```

Result:

PL/SQL procedure successfully completed.

FEEDBACK Procedure

This procedure enables you to provide feedback to Select AI to potentially improve query generation accuracy. You have the option to provide positive or negative feedback, as well as textual comments or revised SQL queries.

Syntax

```

DBMS_CLOUD_AI.FEEDBACK(
    profile_name      IN  VARCHAR2,
    sql_id            IN  DBMS_ID,
    feedback_type     IN  VARCHAR2 DEFAULT NULL,
    response          IN  CLOB DEFAULT NULL,
    feedback_content  IN  CLOB DEFAULT NULL,
    operation         IN  VARCHAR2 DEFAULT 'ADD'
);

```

```

DBMS_CLOUD_AI.FEEDBACK(
    profile_name      IN  VARCHAR2,
    sql_text          IN  CLOB,
    feedback_type     IN  VARCHAR2 DEFAULT NULL,
    response          IN  CLOB DEFAULT NULL,
    feedback_content  IN  CLOB DEFAULT NULL,
    operation         IN  VARCHAR2 DEFAULT 'ADD'
);

```

Parameters

Parameter	Description
profile_name	Specifies the AI profile to use. If you do not provide a profile_name, Select AI uses the default profile set in the session. This is a mandatory parameter.
sql_id	Identifies the SQL query. One sql_id can have only one feedback entry. This is a mandatory parameter.
sql_text	Contains the full text of the SQL query. This is a mandatory parameter.
feedback_type	Specifies the type of feedback. The available values are: - positive: Accept the generated SQL. - negative: Provide the necessary SQL query improvements by identifying errors in the query.
	This is a mandatory parameter when operation is <i>add</i> .
response	Represents the correct SQL query result the user expects. This is a mandatory parameter when operation is <i>add</i> and feedback_type is <i>negative</i> .
feedback_content	Captures the user's natural language feedback. You have the option to use this parameter along with response.
operation	Specifies the operation to perform. The accepted values are: - add (default): Add your feedback by specifying the feedback_type. - delete: Delete your feedback by providing the sql_id

Note

The DBMS_CLOUD_AI.FEEDBACK procedure enables you to specify sql_id or sql_text, therefore, the feedback_type is necessary whereas if you are using the feedback action, the LLM dynamically determines or interprets the feedback type.

Example**Example: Provide Feedback for the Generated SQL Using Add or Delete Operations**

The following example demonstrates using the DBMS_CLOUD_AI.FEEDBACK procedure to accept or improve the generated SQL by specifying the parameters from the procedure.

```
EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',
                           sql_id=> '852w8u83gk1',
                           feedback_type=>'positive',
                           operation=>'add');
EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',
                           sql_text=> 'select ai showsql how many
movies',
                           feedback_type=> 'negative',
                           response=>'SELECT SUM(1) FROM
"ADB_USER"."MOVIES"',
```

```
feedback_content=>'Use SUM instead of  
COUNT');  
EXEC DBMS_CLOUD_AI.FEEDBACK(profile_name=>'OCI_FEEDBACK1',  
                             sql_id=> '852w8u83gk1',  
                             operation=>'delete');
```

Vector Index for FEEDBACK

Select AI creates a default vector index named `<profile_name>_FEEDBACK_VECINDEX` with default attributes when you use the feedback feature for the first time. You can modify its attributes such as `similarity_threshold` and `match_limit` by using the `DBMS_CLOUD_AI.UPDATE_VECTOR_INDEX` procedure. This index helps refine future generated SQL based on the feedback provided. This table is dropped when the associated AI profile is dropped. You can also drop `<profile_name>_FEEDBACK_VECINDEX`. When you do, Select AI no longer uses feedback as hints for the `runsql`, `showsql`, and `explainsql` actions. However, if you submit new feedback using the Select AI feedback feature, Select AI automatically creates a new feedback vector index

 **Note**

The default value of `match_limit` for feedback is 3.

Vector Table Name

The table `<profile_name>_FEEDBACK_VECINDEX$VECTAB` contains vector representations (embeddings) of user feedback along with other parameters, which Select AI uses to improve SQL generation over time.

Parameters

Column	Description
attributes	Includes JSON object attributes as per the FEEDBACK Procedure .
content	Contains the user prompt.
embedding	Contains vector representations (embeddings) of user prompt.

Example

The following example demonstrates using the automatically generated vector index table to query and provide feedback.

```
SQL> select content, attributes from OCI_FEEDBACK1_FEEDBACK_VECINDEX$VECTAB  
where JSON_VALUE(attributes, '$.sql_text') = 'select ai showsql how many  
movies';
```

```
CONTENT  
-----  
-----  
how many movies  
ATTRIBUTES  
-----  
-----
```

```
{ "response": "SELECT SUM(1) FROM
\"ADB_USER\".\"MOVIES\"\", \"feedback_type\": \"negative\", \"sql_id\": null, \"sql_text\":
select ai showsql how many movies\", \"feedback_content\": null}
DBMS_CLOUD_AI.feedback Procedure(Positive Feedback)
```

SUMMARIZE Function

This function summarizes your content based on the customization options you provide as parameters.

Syntax

```
DBMS_CLOUD_AI.SUMMARIZE(
  content          IN CLOB          DEFAULT NULL,
  credential_name  IN VARCHAR2     DEFAULT NULL,
  location_uri     IN VARCHAR2     DEFAULT NULL,
  profile_name     IN VARCHAR2     DEFAULT NULL,
  user_prompt     IN CLOB          DEFAULT NULL,
  params          IN CLOB          DEFAULT NULL
) RETURN CLOB;
```

Parameters

Parameter	Description
content	Specifies the text you want to summarize. Either content or location_uri must be specified. This is not a mandatory parameter.
credential_name	Identifies the credential object used to authenticate with the object store. You must create this credential using DBMS_CLOUD.CREATE_CREDENTIAL.
location_uri	Provides the URI where the text is stored or the path to a local file. Either content or location_uri must be specified. For example: Object storage: https://objectstorage.ca-toronto-1.oraclecloud.com/n/namespace-string/b/bucketname/o/data_folder/summary/test_file.txt Local file: summary_gobject:test_file.txt
profile_name	Specifies the AI profile to use. If you do not provide a profile_name, Select AI uses the default profile set in the session. If no default profile is set, it returns the error: <i>ORA-20046: AI profile is not enabled in the session.</i> The default value is NULL.
user_prompt	Supplies a natural language prompt to guide or customize the summary. You can include additional instructions beyond summary parameters. For example, <i>The summary should start with "The summary of the article is: "</i> This parameter is not mandatory.

Note

Use this parameter only when you provide location_uri.

Parameter	Description
params	Defines summarization parameters. See SUMMARIZE Parameters .

Example

See Example: Select AI Summarize to explore.

SUMMARIZE Parameters

These attributes manage generating summary with custom parameters. Some attributes are optional and have a default value.

Attributes

Attribute Name	Description
	Default Value
min_words	0Specifies the approximate minimum number of words the generated summary is expected to contain.
 Note This parameter acts as a guideline rather than a strict limit: the actual length of the summary may vary depending on the content provided and the model's interpretation.	
max_words	2Specifies the approximate maximum number of words the generated summary is expected to contain. 0
 Note This parameter acts as a guideline rather than a strict limit, the actual length of the summary may vary depending on the content provided and the model's interpretation.	

Attribute Name	Description
	Default Value
summary_style	Specifies the format style for the summary. The following are the available summary format options: - paragraph: the summary is presented in one or more paragraphs. - list: the summary is a list of key points from the text.
chunk_processing_method	When the text exceeds the token limit that the LLM can process, it must be split into manageable chunks. This parameter enables you to choose the method for processing these chunks. The following options are provided: - iterative_refinement. See #unique_579 for more details. - map_reduce. See #unique_1032 for more details.
extractiveness_level	Determines how closely the summary follows the original wording of the input. It controls the degree to which the model extracts versus rephrases it. The following are the options: - High: The summary stays close to the original phrasing, reusing sentences and expressions where possible. - Medium: A balanced mix of extraction and paraphrasing. - Low: Allows more freedom to reword, restructure, or simplify the input, focusing on meaning rather than exact phrasing.

 **Note**

This setting serves as guidance for the model's summarization behavior, it does not enforce a strict rule. The actual style and wording of the summary may vary based on the input content and model decisions.

TRANSLATE Function

This function enables you to translate your text into the specified target_language. You can supply the source_language and the target_language parameters in the function or they can be taken from the user's AI profile. If your AI profile does not include a

source_language attribute, the generative AI provider automatically detects the input language. If the target_language attribute is missing, Select AI returns an error.

Syntax

```
DBMS_CLOUD_AI.TRANSLATE(  
    profile_name      IN VARCHAR2,  
    text              IN CLOB,  
    source_language   IN VARCHAR2 DEFAULT NULL,  
    target_language   IN VARCHAR2 DEFAULT NULL  
) RETURN CLOB;
```

Parameters

Parameter	Description
profile_name	Specifies the AI profile to use. This is not a mandatory parameter.
text	Specifies the text you want to translate. This is a mandatory parameter.
source_language	Language of the input text
target_language	Language into which the text is translated.

Example

See Example: Select AI Translate to explore.

DBMS_CLOUD_AI_AGENT Package

This package defines and manages Select AI agents, tasks, tools, and orchestration.

① Note

Support for the DBMS_CLOUD_AI_AGENT package is available in Oracle Database 19c starting with version 19.29, and in Oracle Database 26ai starting with version 23.26.

Summary of DBMS_CLOUD_AI_AGENT Subprograms

This section covers the Select AI Agent's DBMS_CLOUD_AI_AGENT subprograms provided with Autonomous AI Database.

Subprogram	Description
CREATE_AGENT Procedure	This procedure creates an agent.
CREATE_AGENT Attributes	Provides attributes for creating an agent.
ENABLE_AGENT Procedure	This procedure enables an agent.
DISABLE_AGENT Procedure	This procedure disables an agent.
DROP_AGENT Procedure	This procedure drops an existing agent.

Subprogram	Description
CREATE_TASK Procedure	This procedure creates a task that an agent and agent team can include.
CREATE_TASK Attributes	Provides attributes for creating an agent task.
ENABLE_TASK Procedure	This procedure enables an agent task.
DISABLE_TASK Procedure	This procedure disables an agent task.
DROP_TASK Procedure	This procedure drops an existing agent task.
CREATE_TOOL Procedure	This procedure creates custom tools that an agent can include.
CREATE_TOOL Attributes	Provides attributes for creating tools.
ENABLE_TOOL Procedure	This procedure enables a registered tool.
DISABLE_TOOL Procedure	This procedure disables a registered tool.
DROP_TOOL Procedure	This procedure drops an existing tool.
CREATE_TEAM Procedure	This procedure creates an agent team and includes agents and tasks.
CREATE_TEAM Attributes	Provides attributes for creating an agent team.
SET_TEAM Function	This procedure sets the agent team in the current session.
GET_TEAM Function	This procedure returns the agent team set in the current session.
CLEAR_TEAM Function	This procedure clears the agent team set in the current session.
RUN_TEAM Function	This procedure creates a new agent or runs a paused one.
ENABLE_TEAM Procedure	This procedure enables an agent team.
DISABLE_TEAM Procedure	This procedure disables an agent team.
DROP_TEAM Procedure	This procedure drops an existing agent team.

CREATE_AGENT Procedure

Use the `DBMS_CLOUD_AI_AGENT.CREATE_AGENT` procedure to register a new AI agent in the Select AI Agent framework. Define the agent's identity using `agent_name`, configure its behavior with the `attributes` parameter, and optionally provide a description.

Syntax

```
DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
    agent_name      IN  VARCHAR2,
    attributes      IN  CLOB,
    status         IN  VARCHAR2  DEFAULT NULL,
    description     IN  CLOB      DEFAULT NULL
);
```

Parameters

Parameter	Description
agent_name	A name for the AI agent. The agent name must follow the naming rules of Oracle SQL identifier. Maximum length of name is 125 characters. This is a mandatory parameter.
attributes	Custom attributes for the AI agent. See CREATE_AGENT Attributes for more details.
status	Status of the agent. The default value is enabled.
description	User-specified description for the AI agent. The default value is NULL.

Example

The following example creates an agent to process customer product returns. For a complete example, see Example: Create a Product Return Agent.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
    agent_name => 'CustomerAgent',
    attributes => '{
      "profile_name": "GOOGLE",
      "role": "You are an experienced customer agent who
deals with customers return request."
    }'
  );
END;
```

PL/SQL procedure successfully completed.

CREATE_AGENT Attributes

These attributes define an agent's role and the AI profile name.

Attributes

Attribute Name	Description
	enable
profile_name	- The AI profile that the agent is using to send request to LLM. This is a mandatory parameter.

Attribute Name	Description
	enable
role	- Define the agent's function and provide context to the agent. This is sent to LLM. This is a mandatory parameter.
enable_human_tool	TEnable agent to ask questions to the user for information or clarification. true

The following example shows you how you can customize agent attributes in the procedure.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_AGENT(
    agent_name => 'Customer_Return_Agent',
    attributes => '{
      "profile_name": "GOOGLE",
      "role": "You are an experienced customer agent who
deals with customers return request.",
      "enable_human_tool": "False"
    }'
  );
END;
/
```

PL/SQL procedure successfully completed

ENABLE_AGENT Procedure

This procedure enables the AI agent that the user specifies. The procedure changes the status of the AI agent to ENABLED.

Syntax

```
DBMS_CLOUD_AI_AGENT.ENABLE_AGENT(
  agent_name          IN  VARCHAR2
);
```

Parameters

Parameter	Description
agent_name	Name for the AI agent to enable. This parameter is mandatory.

Example to Enable AI Profile

```
BEGIN
    DBMS_CLOUD_AI_AGENT.ENABLE_AGENT(
        agent_name => 'Customer_Return_Agent'
    );
END;
/
```

DISABLE_AGENT Procedure

This procedure disables the AI agent in the current database. The status of the AI agent is changed to DISABLED by this procedure.

Syntax

```
DBMS_CLOUD_AI_AGENT.DISABLE_AGENT(
    agent_name          IN  VARCHAR2
);
```

Parameters

Parameter	Description
agent_name	Name for the AI agent. This parameter is mandatory.

Example

```
BEGIN
    DBMS_CLOUD_AI_AGENT.DISABLE_AGENT(
        agent_name => 'Customer_Return_Agent'
    );
END;
/
```

DROP_AGENT Procedure

The procedure drops an existing AI agent. If the agent does not exist, then the procedure throws an error.

Syntax

```
DBMS_CLOUD_AI_AGENT.DROP_AGENT(
    agent_name          IN  VARCHAR2,
```

```
        force            IN  BOOLEAN DEFAULT FALSE
    );
```

Parameters

Parameter	Description
agent_name	Name of the AI agent. This is a mandatory parameter.
force	If TRUE, then the procedure ignores errors if AI agent does not exist. The default value for this parameter is FALSE.

Example

```
BEGIN
    DBMS_CLOUD_AI_AGENT.DROP_AGENT(agent_name => 'Customer_Return_Agent');
END;
/
```

Usage Notes

Use force to drop an agent and ignore errors if AI agent does not exist.

CREATE_TASK Procedure

Use the `DBMS_CLOUD_AI_AGENT.CREATE_TASK` procedure to define a task that a Select AI agent can include in its reasoning process. Each task has a unique name and a set of attributes that specify the agent's behavior when planning and performing the task.

You can define dependencies between tasks using the input attribute, allowing Select AI to pass the output of one task into another. This supports chained reasoning and multi-step workflows.

Syntax

```
DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name          IN  VARCHAR2,
    attributes          IN  CLOB,
    status              IN  VARCHAR2  DEFAULT NULL,
    description         IN  CLOB      DEFAULT NULL
);
```

Parameters

Parameter	Description
task_name	A unique name for the task. This is a mandatory parameter.
attributes	JSON object containing task-specific attributes. See CREATE_TASK Attributes .
status	Status of the task. The default value is ENABLED.

Parameter	Description
description	User-defined description to help identify the task. This value is stored in the database but not sent to the LLM. The default value is NULL.

Example

The following example creates a task to process a product return from a customer.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CREATE_TASK(
    task_name => 'Return_And_Price_Match',
    attributes => '{"instruction": "Process a product return request from a
customer:{query}' ||
'1. Ask customer the reason for return (price match or
defective)' ||
'2. If price match:' ||
'  a. Request customer to provide a price match link' ||
'  b. Use websearch tool to get the price for that price
match link' ||
'  c. Ask customer if they want a refund.' ||
'  d. Send email notification only if customer accept
the refund.' ||
'3. If defective:' ||
'  a. Process the defective return",
"tools": ["WebSearch", "Email"]}'
  );
END;
/
```


CREATE_TASK Attributes

These attributes define agent's instructions and tools available for the task.

Attributes

Attribute Name	Description
instruction	- A clear, concise statement that describes what the task should accomplish. This text is included in the prompt sent to the LLM. You can include a {query} placeholder to represent the your natural language input. For example: <pre>SELECT AI AGENT <user_prompt></pre> This is a mandatory parameter.
tools	- A JSON array of tool names that the agent can use to complete the task. For example: "tools": ["RAG", "SQL", "WEBSEARCH", "NOTIFICATION"]. This is a mandatory parameter.
input	- Identifies other tasks whose outputs should be used as input for this task. Select AI automatically provides these outputs to the LLM.
enable_human_tool	- Enable agent to ask questions to user when it requires information or clarification during a task. Default value is true.

Note

Task enable_human_tool overwrites the agent enable_human_tool attribute. If agent enable_human_tool is set to false and task enable_human_tool is set to true, during the task run, the LLM is expected to ask questions.

ENABLE_TASK Procedure

Use the `DBMS_CLOUD_AI_AGENT.ENABLE_TASK` procedure to enable a previously defined task. This changes the task's status to `ENABLED`, allowing the agent to use it during processing.

Syntax

```
DBMS_CLOUD_AI_AGENT.ENABLE_TASK(  
    task_name          IN  VARCHAR2  
);
```

Parameters

Parameter	Description
task_name	Name for the task to enable. This parameter is mandatory.

Usage Notes

The task must already exist in the system. This procedure updates the task status, making it available for agent process during reasoning workflows.

Example to Enable a Task

```
BEGIN  
    DBMS_CLOUD_AI_AGENT.ENABLE_TASK(  
        task_name      => 'Return_And_Price_Match'  
    );  
END;  
/
```

DISABLE_TASK Procedure

Use the `DBMS_CLOUD_AI_AGENT.DISABLE_TASK` procedure to disable a task in the current database. This changes the task's status to `DISABLED`, preventing the agent from using it during processing.

Syntax

```
DBMS_CLOUD_AI_AGENT.DISABLE_TASK(  
    task_name          IN  VARCHAR2  
);
```

Parameters

Parameter	Description
task_name	Name of the task to disable. This parameter is mandatory.

Usage Notes

Disabling a task does not delete it; you can re-enable it later using `DBMS_CLOUD_AI_AGENT.ENABLE_TASK`. Once disabled, the task becomes inactive and the agent no longer considers it during task selection.

Example

The following example disables an existing task.

```
BEGIN
    DBMS_CLOUD_AI_AGENT.DISABLE_TASK(
        task_name => 'Return_And_Price_Match'
    );
END;
/
```

DROP_TASK Procedure

Use the `DBMS_CLOUD_AI_AGENT.DROP_TASK` procedure to remove a previously defined task from the database. If the task does not exist, you can suppress the error by setting the `force` parameter to `TRUE`.

Syntax

```
DBMS_CLOUD_AI_AGENT.DROP_TASK(
    task_name IN VARCHAR2,
    force      IN BOOLEAN DEFAULT FALSE
);
```

Parameters

Parameter	Description
<code>task_name</code>	Name of the task to drop. This is a mandatory parameter.
<code>force</code>	If <code>TRUE</code> , then the procedure ignores errors if the task does not exist. The default value for this parameter is <code>FALSE</code> .

Example

```
BEGIN
    DBMS_CLOUD_AI_AGENT.DROP_TASK(task_name => 'Return_And_Price_Match');
END;
/
```

Usage Notes

Use `force` to drop a task and ignore errors if the task does not exist.

CREATE_TOOL Procedure

Use the `DBMS_CLOUD_AI_AGENT.CREATE_TOOL` procedure to register a custom tool that an agent can use during task processing. Each tool is identified by a unique `tool_name` and includes attributes that define its purpose, implementation logic, and metadata.

The tools can be custom PL/SQL procedures or functions, or built-in tool types such as RAG, SQL, WEBSEARCH, or NOTIFICATION.

Syntax

```
DBMS_CLOUD_AI_AGENT.CREATE_TOOL(  
    tool_name      IN  VARCHAR2,  
    attributes     IN  CLOB,  
    status         IN  VARCHAR2  DEFAULT NULL,  
    description    IN  CLOB      DEFAULT NULL  
);
```

Parameters

Parameter	Description
<code>tool_name</code>	A unique name to identify the tool. This is a mandatory parameter.
<code>attributes</code>	Tool attributes in JSON format. See CREATE_TOOL Attributes . This is a mandatory parameter.
<code>status</code>	Status of the tool. The default value is ENABLED.
<code>description</code>	User-defined description to help identify the tool. This value is stored in the database but not sent to the LLM. The default value is NULL.

Example

The following example creates a tool for to generate an email to the recipient.

```
BEGIN  
    DBMS_CLOUD_AI_AGENT.CREATE_TOOL(  
        tool_name => 'NOTIFICATION',  
        attributes => '{"tool_type": "EMAIL"}'  
    );  
END;  
/
```

CREATE_TOOL Attributes

These attributes provide tool configuration using a JSON object in the attributes parameter. Each attribute configures how the agent uses the tool during reasoning.

Attributes

Attribute Name	Default Value	Description
instruction	-	A clear, concise statement that describes what the tool should accomplish and how to do it. This text is included in the prompt sent to the LLM. This is a mandatory parameter.
function	-	Specifies the PL/SQL procedure or function to call when the tool is used. This is a mandatory parameter for custom tools.
tool_type	-	Specifies a built-in tool type. If set, instruction and function are not required. The possible values are: - SQL - RAG - WEBSEARCH - NOTIFICATION You can create built-in tools with customized names and parameters. You can also create multiple built-in tools with same tool_type but different tool_params.
tool_params	-	Specifies parameters for registering built-in tools.
tool_inputs	-	Describes input arguments. Similar to column comments in a table. For example: <pre>"tool_inputs": [{ "name": "data_guard", "description": "Only supported values are "Enabled" and "Disabled" }]</pre>

Prerequisites for Built-In Tool

Perform these steps before you use built-in tools:

RAG Tool

1. Create an AI profile that references a vector index. See Enhance Natural Language Prompts Using Retrieval Augmented Generation (RAG).

NOTIFICATION Tool

The following are supported: SLACK and EMAIL.

SLACK Tool:

1. Set up a Slack application to receive messages from Autonomous Database.
2. Create a credential using `DBMS_CLOUD_NOTIFICATION` to send Slack notifications from Autonomous Database. See [Send Slack Notifications from Autonomous AI Database](#).

EMAIL Tool:

Note

The email tool sends email notification from OCI.

1. Identify your SMTP endpoint. For example, select one of the SMTP endpoints:

```
smtp.us-phoenix-1.oraclecloud.com
smtp.us-ashburn-1.oraclecloud.com
smtp.email.uk-london-1.oci.oraclecloud.com
smtp.email.eu-frankfurt-1.oci.oraclecloud.com
```

2. Grant SMTP access using an ACL. See [Configuring SMTP Connection](#).
3. Generate SMTP credentials for email delivery. The `UTL_SMTP` package uses credentials to authenticate with email delivery servers when you send an email. See [Generate SMTP Credentials for a User](#) for more information.
4. Create approved senders. Complete this step for all email addresses you use as the From address with `UTL_SMTP.MAIL` subprogram. See [Managing Approved Senders](#) for more information.
5. Grant SMTP access to the user by adding Access Control Entry (ACE). For more details, see [Send Email on Autonomous AI Database](#).

Built-In Tool Types in `DBMS_CLOUD_AI_AGENT` Package

You can register built-in tools using the `tool_type` and `tool_params` attributes.

Tool Type	Description	Mandatory Parameters
SQL	Translates natural language into SQL queries.	profile_name
RAG	Performs Retrieval-Augmented Generation through vector search.	profile_name

Tool Type	Description	Mandatory Parameters
NOTIFICATION	Sends messages to a Slack channel or sends emails using SMTP. Available options: - slack - email	notification_type. When notification_type is slack then credential_name and channel are mandatory. When notification_type is email then credential_name, recipient, sender, and smtp_host are mandatory.
WEBSEARCH	Searches information from the web.	credential_name

ENABLE_TOOL Procedure

Use the `DBMS_CLOUD_AI_AGENT.ENABLE_TOOL` procedure to enable a previously registered tool. This changes the tool's status to `ENABLED`, allowing the agent to use it tasks.

Syntax

```
DBMS_CLOUD_AI_AGENT.ENABLE_TOOL(  
    tool_name          IN VARCHAR2  
);
```

Parameters

Parameter	Description
tool_name	Name for the tool to enable. This parameter is mandatory.

Example

```
BEGIN  
    DBMS_CLOUD_AI_AGENT.ENABLE_TOOL(  
        tool_name => 'SQL'  
    );  
END;  
/
```

DISABLE_TOOL Procedure

Use the `DBMS_CLOUD_AI_AGENT.DISABLE_TOOL` procedure to disable a registered tool. This changes the tool's status to `DISABLED`, preventing the agent from using it in tasks.

Syntax

```
DBMS_CLOUD_AI_AGENT.DISABLE_TOOL(  
    tool_name          IN  VARCHAR2  
);
```

Parameters

Parameter	Description
tool_name	Name of the tool to disable. This parameter is mandatory.

Example

The following example disables an existing tool.

```
BEGIN  
    DBMS_CLOUD_AI_AGENT.DISABLE_TOOL(  
        tool_name      => 'SQL'  
    );  
END;  
/
```

DROP_TOOL Procedure

Use the `DBMS_CLOUD_AI_AGENT.DROP_TOOL` procedure to remove a previously registered tool with the agent. If the task does not exist, you can suppress the error by setting the `force` parameter to `TRUE`.

Syntax

```
DBMS_CLOUD_AI_AGENT.DROP_TOOL(  
    tool_name IN  VARCHAR2,  
    force     IN  BOOLEAN DEFAULT FALSE  
);
```

Parameters

Parameter	Description
tool_name	Name of the tool to drop. This is a mandatory parameter.
force	If <code>TRUE</code> , then the procedure ignores errors if tool does not exist. The default value for this parameter is <code>FALSE</code> .

Example

```
BEGIN  
    DBMS_CLOUD_AI_AGENT.DROP_TOOL(tool_name => 'SQL');
```



```
END;  
/
```

Usage Notes

Use force to drop a tool and ignore errors if the tool does not exist.

CREATE_TEAM Procedure

Use the `DBMS_CLOUD_AI_AGENT.CREATE_TEAM` procedure to define a team of AI agents that work together to accomplish tasks. You specify agents and their associated tasks, along with how the tasks should be processed.

Syntax

```
DBMS_CLOUD_AI_AGENT.CREATE_TEAM(  
    team_name      IN VARCHAR2,  
    attributes      IN CLOB,  
    status          IN VARCHAR2 DEFAULT NULL,  
    description     IN CLOB      DEFAULT NULL  
);
```

Parameters

Parameter	Description
team_name	A unique name to identify the AI agent team. This is a mandatory parameter.
attributes	JSON-formatted string defining team attributes such as agents and process. See CREATE_TEAM Attributes . This is a mandatory parameter.
status	Status of the team. The default value is ENABLED.
description	User-defined description to identify the team's purpose. This value is stored in the database but not sent to the LLM. The default value is NULL.

Example

The following example creates an agent team to process a customer's product return.

```
BEGIN  
    DBMS_CLOUD_AI_AGENT.CREATE_TEAM(  
        team_name =>  
        'ReturnAgency',  
        attributes => '{"agents": [{"name":"Customer_Return_Agent","task" :  
        "Return_And_Price_Match"}],  
        "process":  
        "sequential"}');  
  
END;  
/
```

CREATE_TEAM Attributes

These attributes provide AI agent team configuration. Each team has a unique team_name and uses a set of attributes to define its structure and execution strategy.

Attributes

Attribute Name	Description
agents	- Specifies JSON array listing agents and their corresponding tasks. An agent can be assigned to multiple tasks using the same name. This is a mandatory parameter. Example: <pre>[{"name": "<agent_name>", "task": "<task_name>" }</pre>
process	- Specifies how tasks are run. Use "sequential" for tasks to run one after another in order.

SET_TEAM Function

Use DBMS_CLOUD_AI_AGENT.SET_TEAM to specify the AI agent team for the current session. Once a team is set, only the select ai agent action is enabled, the system does not support other Select AI actions.

Syntax

```
DBMS_CLOUD_AI_AGENT.SET_TEAM(
    team_name      IN  VARCHAR2,
);
```

Parameters

Parameter	Description
team_name	Specifies the name of the AI Agent team to use in the current session. The team must already exist in the system. This parameter is mandatory.

Example

The following example sets the user-specified AI agent team in the current session.

```
EXEC DBMS_CLOUD_AI_AGENT.SET_TEAM('ReturnAgency');
```

GET_TEAM Function

Use `DBMS_CLOUD_AI_AGENT.GET_TEAM` to return the AI agent team for the current session.

Syntax

```
DBMS_CLOUD_AI_AGENT.GET_TEAM RETURN VARCHAR2;
```

Example

The following example returns the AI agent team in the current session.

```
SELECT DBMS_CLOUD_AI_AGENT.GET_TEAM from DUAL;
```

CLEAR_TEAM Function

Use `DBMS_CLOUD_AI_AGENT.CLEAR_TEAM` to clear the AI agent team set in the current session. This procedure does not drop the team.

Syntax

```
DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;
```

Example

The following example clears the AI agent team in the current session.

```
BEGIN
  DBMS_CLOUD_AI_AGENT.CLEAR_TEAM;
END;
/
```

RUN_TEAM Function

Use `DBMS_CLOUD_AI_AGENT.RUN_TEAM` to start a new AI agent team or resume a paused one that is waiting for human input. If you provide an existing process ID and the associated team process is in the `WAITING_FOR_HUMAN` state, the function resumes the workflow using the input you provide as the human response.

Syntax

```
DBMS_CLOUD_AI_AGENT.RUN_TEAM(
  team_name          IN VARCHAR2,
```

```
        user_prompt          IN  CLOB DEFAULT NULL,  
        params               IN  CLOB DEFAULT NULL,  
    );
```

Parameters

Parameter	Description
team_name	Specifies the name of the AI Agent team to use in the current session. The team must already exist in the system. This parameter is mandatory.
user_prompt	Specifies the user query or response provided to the task. If the task is in the RUNNING state, the input acts as a placeholder for the {query} in the task instruction. If the task is in the WAITING_FOR_HUMAN state, the input serves as the human response.
params	Specifies customized parameters used during AI agent team process. Supported parameters include: - conversation_id: Identifies the conversation session associated with the agent team. Use this parameter for stateless application integrations (for example, APEX) to distinguish between different user connections. - variables: Define key-value pairs that provide additional input to the agent team. These variables help customize the team's behavior during the process.

Example

The following example runs the user-specified AI agent team in the current session.

```
l_final_answer := DBMS_CLOUD_AI_AGENT.RUN_TEAM(  
    team_name => 'ReturnAgency',  
    user_prompt => 'I want to return computer desk chair',  
    params => '{"conversation_id": "" || l_conversation_id || ""}'  
);
```

ENABLE_TEAM Procedure

Use the DBMS_CLOUD_AI_AGENT.ENABLE_TEAM procedure to enable a user-specified AI agent team. This changes the team's status to ENABLED.

Syntax

```
DBMS_CLOUD_AI_AGENT.ENABLE_TEAM(  
    team_name          IN  VARCHAR2  
);
```

Parameters

Parameter	Description
team_name	Name for the AI agent team to enable. This parameter is mandatory.

Example

The following example enables a user-specified AI agent team.

```
BEGIN
    DBMS_CLOUD_AI_AGENT.ENABLE_TEAM(
        team_name => 'ReturnAgency'
    );
END;
/
```

DISABLE_TEAM Procedure

Use the `DBMS_CLOUD_AI_AGENT.DISABLE_TEAM` procedure to disable an existing AI agent team. This changes the team's status to `DISABLED`.

Syntax

```
DBMS_CLOUD_AI_AGENT.DISABLE_TEAM(
    team_name          IN  VARCHAR2
);
```

Parameters

Parameter	Description
team_name	Name of the AI agent team to disable. This parameter is mandatory.

Example

The following example disables an existing AI agent team.

```
BEGIN
    DBMS_CLOUD_AI_AGENT.DISABLE_TEAM(
        team_name => 'ReturnAgency'
    );
END;
/
```

DROP_TEAM Procedure

Use the `DBMS_CLOUD_AI_AGENT.DROP_TEAM` procedure to remove a previously created AI agent team.

Syntax

```
DBMS_CLOUD_AI_AGENT.DROP_TEAM(  
    team_name IN VARCHAR2,  
    force      IN BOOLEAN DEFAULT FALSE  
);
```

Parameters

Parameter	Description
team_name	Name of the team to drop. This is a mandatory parameter.
force	If TRUE, then the procedure ignores errors if team does not exist. The default value for this parameter is FALSE.

Example

```
BEGIN  
    DBMS_CLOUD_AI_AGENT.DROP_TEAM(team_name => 'ReturnAgency');  
END;  
/
```

Usage Notes

Use force to drop a team and ignore errors if the team does not exist.

DBMS_CLOUD_FUNCTION Package

The DBMS_CLOUD_FUNCTION package allows you invoke OCI and AWS Lambda remote functions in your Autonomous AI Database as SQL functions.

Note

Support for DBMS_CLOUD_FUNCTION is available in Oracle Database 19c starting with version 19.29, and in Oracle AI Database 26ai starting with version 23.26.

Summary of DBMS_CLOUD_FUNCTION Subprograms

This table summarizes the subprograms included in the DBMS_CLOUD_FUNCTION package.

Table 7-1 DBMS_CLOUD_FUNCTION Subprograms

Subprogram	Description
CREATE_CATALOG Procedure	This procedure creates a catalog.
CREATE_FUNCTION Procedure	This procedure creates functions in a catalog.
DROP_CATALOG Procedure	This procedure drops a catalog and functions created using the catalog.
DROP_FUNCTION Procedure	This procedure drops functions from a catalog.

Table 7-1 (Cont.) DBMS_CLOUD_FUNCTION Subprograms

Subprogram	Description
LIST_FUNCTIONS Procedure	This procedure lists all the functions in a catalog.
SYNC_FUNCTIONS Procedure	This procedure enables adding new functions to the catalog and removing functions that have been deleted from the catalog.

CREATE_CATALOG Procedure

The DBMS_CLOUD_FUNCTION.CREATE_CATALOG procedure creates a catalog in the database. A catalog is a set of functions that creates the required infrastructure to execute subroutines. This procedure is overloaded.

Syntax

```
DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
    credential_name      IN VARCHAR2,
    catalog_name         IN VARCHAR2,
    service_provider     IN VARCHAR2,
    cloud_params         IN CLOB
);
```

Parameters

Parameter	Description
credential_name	Specifies the name of the credential for authentication. This parameter is mandatory.
service_provider	Specifies the type of the service provider. This parameter can have OCI or AWS as a parameter value. This parameter is mandatory.
catalog_name	Specifies the catalog name. This parameter is mandatory.
cloud_params	Provides parameter to the function. For example, Compartment OCID and Regions. This parameter is mandatory.

Examples

```
BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
        credential_name => 'OCI_CREDENTIAL',
        catalog_name    => 'OCI_DEMO_CATALOG',
        service_provider => 'OCI',
        cloud_params     => '{"region_id":"us-phoenix-1",
"compartment_id":"compartment_id"}'
    );
```

```

END;
/

BEGIN
  DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
    credential_name => 'AWS_CREDENTIAL',
    catalog_name    => 'AWS_DEMO_CATALOG',
    service_provider => 'AWS',
    cloud_params    => '{"region_id":"us-phoenix-1"}'
  );
END;
/

```

You must use the overloaded procedure for creating a library with External Procedures. This procedure does not require a credential object. The parameters require specifications of the EXTPROC VM and the shared library that contains the external procedures. See [External Procedures Overview](#).

Syntax

```

DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
  library_name           IN VARCHAR2,
  library_listener_url   IN VARCHAR2,
  library_wallet_dir_name IN VARCHAR2,
  library_ssl_server_cert_dn IN VARCHAR2,
  library_remote_path    IN VARCHAR2
);

```

Parameters

Parameter	Descriptions
library_name	Specifies the name provided to the catalog/library in the database. This parameter is mandatory.
library_listener_url	Specifies the FQDN (fully qualified domain name) of the EXTPROC VM that is hosting the shared library that contains the external procedures written in a third generation language like C/C++, and the Port Number at which the SQL*Net listener running in the EXTPROC VM container is accepting connection requests. The parameter is specified as a string in the format in host_name:port_number format. For example: 'extproc-agent-170798.subnet-name.vcn-name.oraclevcn.com:16000' This parameter is mandatory.
library_wallet_dir_name	Specifies the name of the DIRECTORY in the database where the self-signed wallet has been imported into, and stored.

Parameter	Descriptions
library_ssl_server_cert_dn	Specifies the server certificate Distinguished Name (DN) obtained from the wallet. For example: 'CN=extproc-agent-170798' This parameter is mandatory.
library_remote_path	Specifies the full pathname of the directory in EXTPROC VM, where the library containing the external procedures is stored. For example: '/u01/app/oracle/extproc_libs/helloCextproc.so' This parameter is mandatory.

Examples

```

BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_CATALOG (
        library_name          => 'EXTPROC_LIBRARY',
        library_listener_url  => 'extproc-agent-170798.subnet-name.vcn-
name.oraclevcn.com:16000',
        library_wallet_dir_name => 'EXTPROCWALLETDIR',
        library_ssl_server_cert_dn => 'CN=extproc-agent-170798',
        library_remote_path   => '/u01/app/oracle/extproc_libs/
helloCextproc.so'
    );
END;
/

```

Errors

Error Code	Description
ORA-20000	This error is raised in the following condition: cloud_params value is missing or incorrect parameter values are passed.
ORA-20001	This error is raised in the following condition: The credential referenced in the credential_name does not exist.
ORA-20002	This error is raised in the following condition: This error is raised when the catalog already exists.
ORA-20009	This error is raised when the service provider doesn't exist.

Usage Note

- To create a catalog you must be logged in as the ADMIN user or have privileges on the following:
 - DBMS_CLOUD_OCI_FNC_FUNCTIONS_INVOKE
 - DBMS_CLOUD_OCI_FNC_FUNCTIONS_INVOKE_INVOKE_FUNCTION_RESPONSE_T
 - DBMS_CLOUD
 - Read privilege on USER_CLOUD_FUNCTION
 - Read privilege on USER_CLOUD_FUNCTION_CATALOG

CREATE_FUNCTION Procedure

You can manually create a SQL Function that calls its corresponding cloud function in your catalog. While SYNC_FUNCTIONS automatically retrieves and creates wrappers for all the cloud functions in a catalog, CREATE_FUNCTION enables you to create functions with custom return types and response handlers. The CREATE_FUNCTION Procedure is applicable only for Cloud Functions.

The DBMS_CLOUD_FUNCTION.CREATE_FUNCTION procedure is only supported for cloud functions.

Syntax

```
DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    credential_name    IN VARCHAR2,
    catalog_name       IN VARCHAR2,
    function_name      IN VARCHAR2,
    function_id        IN VARCHAR2,
    input_args         IN CLOB      DEFAULT NULL,
    return_type        IN VARCHAR2  DEFAULT 'CLOB',
    response_handler    IN VARCHAR2  DEFAULT NULL
);
```

The response_handler must be defined in this format:

```
<USER DEFINED TYPE> response_handler_name(function_response in CLOB)RETURNS
CLOB;
```

The return value of the response_handler is of <User Defined Type> and the response_json is a JSON document with two fields, in the following format:

```
'{
  "status":"<response status>",
  "response_body":"<response from the function>"
}'
```

Parameters

Parameter	Description
credential_name	Specifies the name of the credential for authentication. This parameter is mandatory.
catalog_name	Specifies the catalog name. This parameter is mandatory.
function_name	Specifies the PL/SQL function name. This parameter is mandatory.
function_id	The function_id parameter value refers to the OCI function or AWS Lambda. This parameter is mandatory.
input_args	Specifies the key value JSON pair accepting input arguments and their types.

Parameter	Description
return_type	Defines the return type of the function. The return type is of CLOB data type.
response_handler	Specifies the user defined callback to handle response.

Errors

Error Code	Description
ORA-20001	This error is raised when the credential referenced in the credential_name does not exist.
ORA-20003	This error is raised when the specified catalog does not exist.
ORA-20004	This error is raised when the specified function already exists.
ORA-20005	This error is raised when the function ID or function Amazon Resource Names (ARN) does not exist.
ORA-20006	This error is raised when the input arguments are invalid.
ORA-20007	This error is raised when the return type is missing or invalid.
ORA-20008	This error is raised when the response handler is missing or invalid.

Example

The simple case with just input parameters to the function:

```
VAR function_args CLOB;
EXEC :function_args := TO_CLOB('{ "command": "VARCHAR2", "value":
"VARCHAR2"}');
BEGIN
    DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
        credential_name => 'DEFAULT_CREDENTIAL',
        catalog_name    => 'OCI_DEMO_CATALOG',
        function_name    => 'demo_function',
        function_id      =>
'ocid1.fnfunc.oc1.phx.aaaaaaaazkrbjv6ntowwxlbbp5ct4otsj4o2hdw4ayosyosv4sthmya2
lyza',
        input_args       => :function_args);
    );
END;
/
```

Example with return type and response handler:

```
VAR input_param CLOB;
VAR l_return_type VARCHAR2(100);
VAR l_response_handler VARCHAR2(1000);

-- Define function parameters
exec :input_param      := TO_CLOB('{ "command": "VARCHAR2", "value":
"VARCHAR2"}');
```

PL/SQL procedure successfully completed.

```

exec :l_return_type      := 'fintech_rt';

PL/SQL procedure successfully completed.

exec :l_response_handler := 'fintech_response_handler';

PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    credential_name => 'OCI_CRED',
    catalog_name   => 'OCI_DEMO_CATALOG',
    function_name   => 'FINTECH_FUNCTION',
    function_id     => 'ocid1.fnfunc.oc1.phx.aaabbbccccc_example',
    input_args      => :input_param,
    return_type     => :l_return_type,
    response_handler => :l_response_handler
  );
END;
/

```

Next, use this type and response handler during the manual creation of the SQL Wrapper Function.

```

VAR input_param CLOB;
VAR l_return_type VARCHAR2(100);
VAR l_response_handler VARCHAR2(1000);

-- Define function parameters
exec :input_param      := TO_CLOB('{"command": "VARCHAR2", "value":
"VARCHAR2"}');

PL/SQL procedure successfully completed.

exec :l_return_type      := 'fintech_rt';

PL/SQL procedure successfully completed.

exec :l_response_handler := 'fintech_response_handler';

PL/SQL procedure successfully completed.

BEGIN
  DBMS_CLOUD_FUNCTION.CREATE_FUNCTION (
    credential_name => 'OCI_CRED',
    catalog_name   => 'OCI_DEMO_CATALOG',
    function_name   => 'FINTECH_FUNCTION',
    function_id     => 'ocid1.fnfunc.oc1.phx.aaabbbccccc_example',
    input_args      => :input_param,
    return_type     => :l_return_type,
    response_handler => :l_response_handler
  );
END;
/

```

Once the function is created, you can DESCRIBE the function to get its return details.

```
DESC fintech_fun
COLUMN STATUS format a30
COLUMN OUTPUT format a30
```

You can then invoke the function, providing values for the input parameters

```
SET SERVEROUTPUT ON

DECLARE
    l_comp fintech_rt;
BEGIN
    l_comp := fintech_fun(command=>'tokenize',value => 'PHI_INFORMATION');

    DBMS_OUTPUT.put_line ('Status of the function   =  '|| l_comp.status);
    DBMS_OUTPUT.put_line ('Response of the function =  '|| l_comp.output);
END;
/
```

This invokes the fintech_fun cloud function by calling the function reference oocid1.funfn.oci.phx.aaaaaa_example in the OCI_DEMO_CATALOG catalog.

Usage Notes

For invoking External Procedures, you can use the same syntax that is used for User Defined Functions. For example, if you have created a library called EXTPROC_LIBRARY in your database using the CREATE_CATALOG Procedure for External Procedures, you can define a SQL function as shown in this example, and invoke it from a SQL statement.

```
SQL> CREATE OR REPLACE FUNCTION HELLOEXTPROC RETURN VARCHAR2 AS
    LANGUAGE C
    LIBRARY EXTPROC_LIBRARY
    NAME "helloCextproc";
/
```

Function created.

```
SQL> SELECT HELLOEXTPROC() FROM dual;
```

```
HELLOEXTPROC()
-----
-----
Hello C Extproc from FQDN: extproc-agent-170798.subnet-name.vcn-
name.oraclevcn.com
```

DROP_CATALOG Procedure

The `DBMS_CLOUD_FUNCTION.DROP_CATALOG` procedure drops the catalog and functions created using the catalog.

Syntax

```
DBMS_CLOUD_FUNCTION.DROP_CATALOG (  
    catalog_name      IN VARCHAR2  
);
```

Parameters

Parameter	Description
catalog_name	Specifies the catalog name. This parameter is mandatory.

Errors

Error Code	Description
ORA-20003	This error is raised when the specified catalog doesn't exist.

Example:

```
BEGIN  
    DBMS_CLOUD_FUNCTION.DROP_CATALOG (  
        catalog_name      => 'OCI_DEMO_CATALOG'  
    );  
END;  
/
```

DROP_FUNCTION Procedure

The `DBMS_CLOUD_FUNCTION.DROP_FUNCTION` procedure drops the function.

The `DBMS_CLOUD_FUNCTION.DROP_FUNCTION` procedure is only supported for cloud functions.

Syntax

```
DBMS_CLOUD_FUNCTION.DROP_FUNCTION (  
    catalog_name      IN VARCHAR2,  
    function_name     IN VARCHAR2  
);
```

Parameters

Parameter	Description
catalog_name	Specifies the catalog name. This parameter is mandatory.

Parameter	Description
function_name	Specifies the name of the function to be dropped. This parameter is mandatory.

Errors

Error Code	Description
ORA-20003	This error is raised when the specified function does not exist.

Examples

```
BEGIN
    DBMS_CLOUD_FUNCTION.DROP_FUNCTION (
        catalog_name => 'OCI_DEMO_CATALOG',
        function_name => 'demo_function');
END;
/
```

LIST_FUNCTIONS Procedure

This procedure lists all the functions in a catalog.

Note

The LIST_FUNCTIONS Procedure is applicable only for Cloud Functions.

Syntax

```
DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS (
    credential_name IN VARCHAR2,
    catalog_name    IN VARCHAR2,
    function_list   OUT VARCHAR2
);
```

Parameters

Parameter	Description
credential_name	Specifies the name of the credential for authentication. This parameter is mandatory.
function_list	Returns the list of functions in JSON format. This parameter is mandatory.
catalog_name	Specifies the catalog name. This parameter is mandatory.

Errors

Error Code	Description
ORA-20000	This error is raised when incorrect parameter values are passed.
ORA-20001	This error is raised when the credential referenced in the <code>credential_name</code> does not exist.
ORA-20003	This error is raised when the specified catalog does not exist.

Example:

```
VAR function_list CLOB;
BEGIN
    DBMS_CLOUD_FUNCTION.LIST_FUNCTIONS (
        credential_name => 'DEFAULT_CREDENTIAL',
        catalog_name    => 'OCI_DEMO_CATALOG',
        function_list    => :function_list);
);
END;
/
SELECT JSON_QUERY(:function_list, '$' RETURNING VARCHAR2(32676) pretty) AS
search_results FROM dual;
```

SYNC_FUNCTIONS Procedure

This procedure enables adding new functions to the catalog and removing functions that have been deleted from the catalog.

Note

The SYNC_FUNCTIONS Procedure is applicable only for Cloud Functions.

Syntax

```
DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS (
    catalog_name    IN VARCHAR2,
    refresh_rate    IN VARCHAR2 DEFAULT 'DAILY'
);
```

Parameters

Parameter	Description
<code>catalog_name</code>	Specifies the catalog name. This parameter is mandatory.

Parameter	Description
refresh_rate	Specifies the refresh rate of the function. refresh_rate can accept the following values: • HOURLY • DAILY • WEEKLY • MONTHLY The default value for this parameter is DAILY.

Errors

Error Code	Description
ORA-20003	This error is raised when the specified catalog does not exist.
ORA-20004	This error is raised when an invalid value is passed for the refresh_rate parameter.

Example:

```

BEGIN
    DBMS_CLOUD_FUNCTION.SYNC_FUNCTIONS (
        catalog_name => 'OCI_DEMO_CATALOG'
    );
END;
/

```

DBMS_CLOUD_NOTIFICATION Package

The DBMS_CLOUD_NOTIFICATION package allows you to send messages or the output of a SQL query to a provider.

The supported providers with DBMS_CLOUD_NOTIFICATION are: Email, Microsoft Teams, OCI Notifications (ONS), and Slack.

When you use the OCI Notifications service with DBMS_CLOUD_NOTIFICATION package, only the DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE procedure is supported (the DBMS_CLOUD_NOTIFICATION.SEND_DATA procedure is not supported).

Prerequisites

- [Oracle Public Cloud or Multicloud](#)
- [Exadata Cloud@Customer](#)

Oracle Public Cloud or Multicloud

An outbound connectivity must have been configured using a NAT gateway, by your fleet administrator as described below:

- Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
- After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 1. Go to the **Subnet Details** page for the subnet.
 2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
 3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0
 - **Target Type:** NAT Gateway
 - **Target:** The name of the NAT gateway you just created in the VCNIf such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.
 4. Return to the **Subnet Details** page for the subnet.
 5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
 6. In the side menu, under **Resources**, click **Egress Rules**.
 7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** AllIf such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

Summary of DBMS_CLOUD_NOTIFICATION Subprograms

This table summarizes the subprograms included in the package.

Subprogram	Description
SEND_DATA Procedure	Send SQL query output to a provider.
SEND_MESSAGE Procedure	Send a text message to a provider.

SEND_DATA Procedure

The SEND_DATA procedure sends the results of the specified query to a provider.

Syntax

```
DBMS_CLOUD_NOTIFICATION.SEND_DATA(  
    provider          IN  VARCHAR2,  
    credential_name   IN  VARCHAR2,  
    query             IN  CLOB,  
    params            IN  CLOB  
);
```

Parameters

Parameter	Description
provider	Specifies the provider. Valid values are: 'email', 'msteams', and 'slack' This parameter is mandatory.
credential_name	The name of the credential object to access the provider. For the email provider, the credential is the name of the credential of the SMTP approved sender that contains its username and password. For the msteams provider, the credential is the name of the credential. For the Slack provider, the credential's username must be "SLACK_TOKEN" and the password is the Slack Bot Token. This parameter is mandatory.
query	Specifies the SQL query to run. Results are sent to the provider. This parameter is mandatory.

Parameter	Description
params	Specifies specific parameters for the provider in JSON format. For the provider type <code>email</code>, the valid parameters are: <code>sender</code>. This specifies the Email ID of the approved sender in a <code>String</code> value. <code>smtp_host</code>. This specifies the SMTP host name in a <code>String</code> value. <code>subject</code>. This specifies the subject of the email in a <code>String</code> value. The maximum size is 100 characters. <code>recipient</code>. This specifies the email IDs of recipients in a <code>String</code> value. Use a comma between email IDs when there are multiple recipients. <code>to_cc</code>. This specifies the email IDs that are receiving a CC of the email. It is a <code>String</code> value. Use a comma between email IDs when there are multiple CC recipients. <code>to_bcc</code>. This specifies the email IDs that are receiving a BCC of the email. It is a <code>String</code> value. Use a comma between email IDs when there are multiple BCC recipients. <code>message</code>. This specifies the message text in a <code>String</code> value. <code>type</code>. This specifies the output format in a <code>String</code> value as either CSV or JSON. <code>title</code>. This specifies the title of the attachment of SQL output in a <code>String</code> value. The title should only contain letters, digits, underscores, hyphens, or dots as characters in its value due to it being used to generate a file name. For the provider type <code>msteams</code>, the valid parameters are: <code>tenant</code>. This specifies the tenant ID in a <code>String</code> format. <code>team</code>. This specifies the team ID in a <code>String</code> format. <code>channel</code>. This specifies the channel ID in a <code>String</code> format. <code>title</code>. This specifies the title of the file in a <code>String</code> format. The maximum size is 50 characters. <code>type</code>. This specifies the output format in a <code>String</code> value. Valid values are CSV or JSON. For the provider type <code>slack</code>, the valid parameters are: <code>channel</code>. This specifies the Channel ID in a <code>String</code> value. The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the "About" tab. <code>type</code>. This specifies the output format in a <code>String</code> value. Valid values are CSV or JSON. This parameter is mandatory.

Usage Notes

Use the procedure `DBMS_CLOUD.CREATE_CREDENTIAL(credential_name, username, password)` to create the credential object. See `CREATE_CREDENTIAL` Procedure for more information.

- For the `email` provider, the user requires an SMTP connection endpoint for the Email Delivery server to obtain `smtp_host`. The user also requires an SMTP approved sender and its credentials to authenticate with the Email Delivery server to obtain the `credential_name`. SMTP connection must be configured, and SMTP credentials must be generated and approved.
- For the `msteams` provider, the user requires the Microsoft Teams App and a bot configured in it. The app needs to be published to the org and installed after obtaining approval from the admin from the admin center. The user also requires access for `Files.ReadWrite.All`

and ChannelSettings.Read.All permission for Graph API from their Azure Portal. To generate the required token, the user requires bot_id in the username and bot_secret in the password. The maximum file size supported when using DBMS_CLOUD_NOTIFICATION.SEND_DATA for Microsoft Teams is 4MB.

- For the slack provider, the username value can be any valid string and the password is the Slack bot token.

Example

Send SQL output with the email provider:

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

BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_DATA(
    provider          => 'email',
    credential_name   => 'EMAIL_CRED',
    query             => 'SELECT tablespace_name FROM dba_tablespaces',
    params            => json_object('recipient' value 'mark@oracle.com,
suresh@oracle.com',
                                   'to_cc' value 'nicole@oracle.com1,
jordan@oracle.com',
                                   'to_bcc' value 'manisha@oracle.com',
                                   'subject' value 'Test subject',
                                   'type' value 'json',
                                   'title' value 'mytitle',
                                   'message' value 'This is the message',
                                   'smtp_host' value
'smtp.email.example.com',
                                   'sender' value
'approver_sender@oracle.com' )
  );
END;
/
```

Send SQL output with the msteams provider:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(credential_name => 'TEAMS_CRED',
    username        => 'bot_id',
    password        => 'bot_secret');
END;
/

BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(provider => 'msteams',
    credential_name => 'TEAMS_CRED',
    query           => 'SELECT tablespace_name FROM dba_tablespaces',
    params          => json_object('tenant' value '5b743bc*****c0286',
```

```

                                'team'value '0ae401*****5d2bd',
                                'channel'value
'19%3a94be023*****%40thread.tacv2',
                                'title'value 'today',
                                'type'value 'csv')));
END;
/

```

Send SQL output with the slack provider:

```

BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'SLACK_CRED',
    username       => 'SLACK_TOKEN',
    password       => 'password');
END;
/
BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_DATA(
    provider       => 'slack',
    credential_name => 'SLACK_CRED',
    query          => 'SELECT username, account_status, expiry_date FROM
USER_USERS WHERE rownum < 5',
    params         => json_object('channel' value 'C0...08','type' value
'csv'));
END;
/

```

SEND_MESSAGE Procedure

The procedure sends a text message to a provider.

Syntax

```

DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
  provider          IN  VARCHAR2,
  credential_name   IN  VARCHAR2,
  message           IN  CLOB,
  params            IN  CLOB
);

```

Parameters

Parameter	Description
provider	Specifies the provider. Valid values are: 'email', 'msteams', 'slack', and 'oci'. This parameter is mandatory.

Parameter	Description
<code>credential_name</code>	The name of the credential object to access the provider. For the email provider, the credential is the name of the credential of the SMTP approved sender that contains its username and password. For the msteams provider, the credential must contain the <code>bot_id</code> and the <code>bot_secret</code> key in both the username and the password. For the Slack provider, the credential's username must be "SLACK_TOKEN" and the password is the Slack Bot Token. For the oci provider, the credential is a valid Oracle Cloud Infrastructure signing key based credential. This parameter is mandatory.
<code>message</code>	Specifies the message text. This parameter is mandatory.
<code>params</code>	Specifies specific parameters for the provider in JSON format. For the provider type <code>email</code>, the valid parameters are: <code>sender</code>. This specifies the Email ID of the approved sender in a <code>String</code> value. <code>smtp_host</code>. This specifies the SMTP host name in a <code>String</code> value. <code>subject</code>. This specifies the subject of the email in a <code>String</code> value. The maximum size is 100 characters. <code>recipient</code>. This specifies the email IDs of recipients in a <code>String</code> value. Use a comma between email IDs when there are multiple recipients. <code>to_cc</code>. This specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients. <code>to_bcc</code>. This specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients. For the provider type <code>msteams</code>, the valid parameter is: <code>channel</code>. This specifies the Channel ID in a <code>String</code> value. For the provider type <code>slack</code>, the valid parameter is: <code>channel</code>. This specifies the Channel ID in a <code>String</code> value. For the provider OCI Notifications with the type <code>oci</code>, the valid parameters are: <code>topic_ocid</code>: Specifies <code>title</code>: Specifies for the subject line of the message. With the <code>oci</code> provider, there are different possible endpoints for a message, based on the subscriptions. The <code>title</code> parameter is treated differently for different subscriptions: <code>email</code>: The <code>title</code> specifies the subject line of the message. <code>Slack</code>: The <code>title</code> value is not used. If it is included the value is ignored. The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the "About" tab. This parameter is mandatory.

Usage Notes

Use the procedure `DBMS_CLOUD.CREATE_CREDENTIAL(credential_name, username, password)` to create the credential object. See `CREATE_CREDENTIAL` Procedure for more information.

- For the email provider, the user requires an SMTP connection endpoint for the Email Delivery server to obtain `smtp_host`. The user also requires an SMTP approved sender and its credentials to authenticate with the Email Delivery server to obtain the

credential_name. SMTP connection must be configured, and SMTP credentials must be generated and approved.

- For the msteams provider, the user requires the Microsoft Teams App and a bot configured in it. The app needs to be published to the org and installed after obtaining approval from the admin from the admin center. The user also requires access for Files.ReadWrite.All and ChannelSettings.Read.All permission for Graph API from their Azure Portal. To generate the required token, the user requires bot_id in the username and bot_secret in the password.
- For the slack provider, the username value can be any valid string and the password is the Slack bot token.
- The DBMS_CLOUD_NOTIFICATION supports the OCI Notifications provider. See [Overview of Notifications](#) for details on the OCI Notifications provider.
 - To use the OCI Notifications provider, you must do the following:
 - * Create a topic. See [Creating a Topic](#) for details on creating a topic
 - * Create the required subscriptions. See [Creating a Subscription](#) for details on creating a subscription.
 - * After you create the subscriptions, confirm the subscriptions. Messages are only published to confirmed subscriptions. See [Confirming a Subscription](#) for details on confirming subscriptions.
 - The Autonomous AI Database instance, the credential (if you are using user credentials), and the subscription topic must all be located in the same region.
 - Note the following limitations:
 - * Message Size Limit: The message size limit per request is 64KB. Messages exceeding this size cannot be sent or processed within a single request.
 - * Message Delivery Rate Limit: There are different message delivery rate limits depending on the protocol used.
 - * email protocol: the limit is 10 messages per minute per endpoint.

This restriction may impact the speed at which messages can be transmitted and received.
 - * Transactions Per Minute (TPM) per-Tenancy Limit: There is a TPM limit of 60 per topic for this operation. This limit represents the maximum number of messages per minute that can be processed or handled for a given topic. Exceeding this limit may result in messages not being processed or processed at a slower rate.

Examples

Send a text message with the email provider:

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

BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
    provider       => 'email',
```



```

        credential_name => 'EMAIL_CRED',
        message         => 'Subject content',
        params          => json_object('recipient' value 'mark@oracle.com,
suresh@oracle.com',
                                     'to_cc' value 'nicole@oracle.com,
jordan@oracle.com',
                                     'to_bcc' value 'manisha@oracle.com',
                                     'subject' value 'Test subject',
                                     'smtp_host' value
'smtp.email.example.com',
                                     'sender' value
'approver_sender@oracle.com' )
    );
END;
/

```

Send a text message with the msteams provider:

```

BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(credential_name => 'TEAMS_CRED',
        username      => 'bot_id',
        password      => 'bot_secret');
END;
/

BEGIN
    DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider      => 'msteams',
        credential_name => 'TEAMS_CRED',
        message       => 'text from new teams api',
        params        => json_object('channel' value
'C0....08'),'region' value 'india');
END;
/

```

Send a text message with the slack provider:

```

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

BEGIN
    DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider      => 'slack',
        credential_name => 'SLACK_CRED',
        message       => 'Send text from Autonomous AI Database.',
        params        => json_object('channel' value 'C0....08'));
END;
/

```

Send a text message with the OCI Notifications, type oci provider:

```
BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
    provider      => 'oci',
    credential_name => 'OCI_CRED',
    message       => 'Text message for you.',
    params        => json_object('topic_ocid' value 'oci*****pa',
                                'title' value 'Title for message subject' );
END;
```

DBMS_CLOUD_PIPELINE Package

The DBMS_CLOUD_PIPELINE package allows you to create data pipelines for loading and exporting data in the cloud. This package supports continuous incremental data load of files in object store into the database. DBMS_CLOUD_PIPELINE also supports continuous incremental export of table data or query results from the database to object store based on a timestamp column.

Prerequisites

As a developer, you can use DBMS_CLOUD procedures with Autonomous AI Databases deployed on Oracle Public Cloud, Multicloud or Exadata Cloud@Customer.

Depending on the deployment choice, the following prerequisites must be met to use the DBMS_CLOUD procedures with Amazon S3, Azure Blob Storage, and Google Cloud Storage service providers.

-
- [Oracle Public Cloud or Multicloud](#)
 - [Exadata Cloud@Customer](#)

Oracle Public Cloud or Multicloud

An outbound connectivity must have been configured using a NAT gateway, by your fleet administrator as described below:

- Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous AI Database resources reside by following the instructions in [Create a NAT Gateway](#) in *Oracle Cloud Infrastructure Documentation*.
- After creating the NAT gateway, add a route rule and an egress security rule **to each subnet** (in the VCN) where Autonomous AI Database resources reside so that these resources can use the gateway to obtain a public key from your Azure AD instance:
 1. Go to the **Subnet Details** page for the subnet.
 2. In the **Subnet Information** tab, click the name of the subnet's **Route Table** to display its **Route Table Details** page.
 3. In the table of existing **Route Rules**, check whether there is already a rule with the following characteristics:
 - **Destination:** 0.0.0.0/0

- **Target Type:** NAT Gateway
- **Target:** The name of the NAT gateway you just created in the VCN

If such a rule does not exist, click **Add Route Rules** and add a route rule with these characteristics.

4. Return to the **Subnet Details** page for the subnet.
5. In the subnet's **Security Lists** table, click the name of the subnet's security list to display its **Security List Details** page.
6. In the side menu, under **Resources**, click **Egress Rules**.
7. In the table of existing **Egress Rules**, check whether there is already a rule with the following characteristics:
 - **Destination Type:** CIDR
 - **Destination:** 0.0.0.0/0
 - **IP Protocol:** TCP
 - **Source Port Range:** 443
 - **Destination Port Range:** All

If such a rule does not exist, click **Add Egress Rules** and add an egress rule with these characteristics.

Exadata Cloud@Customer

The HTTP Proxy settings in your environment must allow the database to access the cloud service provider.

These settings are defined by your fleet administrator while creating the Exadata Cloud@Customer infrastructure as described in [Using the Console to Provision Exadata Database Service on Cloud@Customer](#).

Note

The network configuration including the HTTP Proxy can only be edited until the Exadata Infrastructure is in **Requires Activation** state. Once it is activated, you cannot edit those settings.

Setting up an HTTP Proxy for an already provisioned Exadata Infrastructure needs a Service Request (SR) in My Oracle Support. See [Create a Service Request in My Oracle Support](#) for details.

Summary of DBMS_CLOUD_PIPELINE Subprograms

This table summarizes the subprograms included in the DBMS_CLOUD_PIPELINE package.

Subprogram	Description
CREATE_PIPELINE Procedure	Creates a new data pipeline.
DROP_PIPELINE Procedure	Drops an existing data pipeline.

Subprogram	Description
RESET_PIPELINE Procedure	Resets the tracking state of a data pipeline. Use reset pipeline to restart the pipeline from the initial state of data load or export. Optionally reset pipeline can purge data in the database or in object store, depending on the type of pipeline.
RUN_PIPELINE_ONCE Procedure	Performs an on-demand run of the pipeline in the current foreground session, instead of a scheduled job.
SET_ATTRIBUTE Procedure	Sets pipeline attributes. There are two overloaded procedures, one to set a single attribute and another to set multiple attributes using a JSON document of attribute name/value pairs
START_PIPELINE Procedure	Starts the data pipeline. When a pipeline is started, the pipeline operation will continuously run in a scheduled job according to the "interval" configured in pipeline attributes.
STOP_PIPELINE Procedure	Stops the data pipeline. When a pipeline is stopped, no future jobs are scheduled for the pipeline.

CREATE_PIPELINE Procedure

The procedure creates a new data pipeline.

Syntax

```
DBMS_CLOUD_PIPELINE.CREATE_PIPELINE(
    pipeline_name      IN   VARCHAR2,
    pipeline_type      IN   VARCHAR2,
    attributes         IN   CLOB          DEFAULT NULL,
    description        IN   VARCHAR2     DEFAULT NULL
);
```

Parameters

Parameter	Description
pipeline_name	Specifies a name for the pipeline. The pipeline name must follow the naming rules of Oracle SQL identifiers. See <i>Identifiers</i> in Oracle Database Database PL/SQL Language Reference or Oracle Database 26ai Database PL/SQL Language Reference for more information. This parameter is mandatory.
pipeline_type	Specifies the pipeline type. Valid values: LOAD, EXPORT This parameter is mandatory.
attributes	Pipeline attributes in JSON format. Default value: NULL See DBMS_CLOUD_PIPELINE Attributes for more information.
description	Description for the pipeline. Default value: NULL

DROP_PIPELINE Procedure

The procedure drops an existing data pipeline. If a pipeline has been started, then it must be stopped before it can be dropped.

Syntax

```
DBMS_CLOUD_PIPELINE.DROP_PIPELINE(  
    pipeline_name      IN   VARCHAR2,  
    force              IN   BOOLEAN DEFAULT FALSE  
);
```

Parameters

Parameter	Description
pipeline_name	Specifies a pipeline name. This parameter is mandatory.
force	Forcibly drop a pipeline, even if it is in started state. Valid values: TRUE, FALSE Default value: FALSE

Usage Note

- In order to drop a pipeline that is in started state, set the force parameter to TRUE.

RESET_PIPELINE Procedure

Resets the tracking state of a data pipeline. Use reset pipeline to restart the pipeline from the initial state of data load or export. Optionally reset pipeline can purge data in the database or in object store, depending on the type of pipeline. A data pipeline must be in stopped state to reset it.

Syntax

```
DBMS_CLOUD_PIPELINE.RESET_PIPELINE(  
    pipeline_name      IN   VARCHAR2,  
    purge_data         IN   BOOLEAN DEFAULT FALSE  
);
```

Parameters

Parameter	Description
pipeline_name	Specifies a name for the pipeline. This parameter is mandatory.

Parameter	Description
purge_data	Purge data applies for either a load pipeline or an export pipeline: - For a load pipeline, when TRUE, truncate the data in database table. - For an export pipeline, when TRUE, delete the files in the object store location. Valid values: TRUE, FALSE Default value: FALSE

Usage Notes

- A data pipeline must be in stopped state to reset it. See [STOP_PIPELINE Procedure](#) for more information.
- For a load pipeline, resetting the pipeline clears the record of the files being loaded by the pipeline. When you call either `START_PIPELINE` or `RUN_PIPELINE_ONCE` after resetting a load pipeline, the pipeline repeats the data load and includes all the files present in the object store location.

When `purge_data` is set to TRUE, `DBMS_CLOUD_PIPELINE.RESET_PIPELINE` does the following:

- Truncates the data in the pipeline's database table you specify with `table_name` attribute.
- Drops the pipeline's status table, and the pipeline's bad file table and error table (if they exist).
- For an export pipeline, resetting the pipeline clears the last tracked data in the database table. When you call either `START_PIPELINE` or `RUN_PIPELINE_ONCE` after resetting an export pipeline, the pipeline repeats exporting data from the table or query.

When `purge_data` set to TRUE, `DBMS_CLOUD_PIPELINE.RESET_PIPELINE` deletes existing files in the object store location specified with the `location` attribute.

RUN_PIPELINE_ONCE Procedure

This procedure performs an on-demand run of the pipeline in the current foreground session, instead of a running in a scheduled job. Use `DBMS_CLOUD_PIPELINE.RUN_PIPELINE_ONCE` to test a pipeline before you start the pipeline as a continuous job.

Syntax

```
DBMS_CLOUD_PIPELINE.RUN_PIPELINE_ONCE(  
    pipeline_name IN VARCHAR2  
);
```

Parameters

Parameter	Description
pipeline_name	Specifies a name for the pipeline to run. This parameter is mandatory.

Usage Notes

- After you perform a test run of a pipeline you can reset the pipeline state using `DBMS_CLOUD_PIPELINE.RESET_PIPELINE`. This allows you to reset the pipeline state before you start the pipeline in a scheduled job.
- If a pipeline is in the started state, then it cannot be run in the foreground session.

SET_ATTRIBUTE Procedure

This procedure sets pipeline attributes. There are two overloaded procedures, one to set a single attribute and another to set multiple attributes using a JSON document of attribute name/value pairs.

Syntax

```
PROCEDURE DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(  
    pipeline_name      IN   VARCHAR2,  
    attribute_name     IN   VARCHAR2,  
    attribute_value    IN   CLOB  
);  
  
PROCEDURE DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE(  
    pipeline_name      IN   VARCHAR2,  
    attributes         IN   CLOB  
);
```

Parameters

Parameter	Description
pipeline_name	Specifies a name for the pipeline to set attributes. This parameter is mandatory.
attribute_name	Specifies the attribute name for the attribute to be set. See DBMS_CLOUD_PIPELINE Attributes for more information.
attribute_value	Specifies the value for the pipeline attribute to set. See DBMS_CLOUD_PIPELINE Attributes for more information.
attributes	Specifies a JSON document containing attribute names and values. See DBMS_CLOUD_PIPELINE Attributes for more information.

Usage Note

- When you use `DBMS_CLOUD_PIPELINE.SET_ATTRIBUTE` to set multiple attributes with the `attributes` parameter, all the existing attributes are deleted and overwritten with the specified attributes from the JSON document.

START_PIPELINE Procedure

The procedure starts the data pipeline. When a pipeline is started, the pipeline operation runs continuously in a scheduled job according to the interval configured with the pipeline attributes.

Syntax

```
DBMS_CLOUD_PIPELINE.START_PIPELINE(  
    pipeline_name      IN    VARCHAR2,  
    start_date         IN    TIMESTAMP WITH TIME ZONE DEFAULT NULL  
);
```

Parameters

Parameter	Description
pipeline_name	Specifies a name for the pipeline. This parameter is mandatory.
start_date	Specifies the starting date for the pipeline job. Default value: NULL

Usage Notes

- By default, a pipeline job begins immediately, as soon as the pipeline is started. To start a pipeline job at a later time specify a valid date or timestamp using the `start_date` parameter.
- See [DBMS_CLOUD_PIPELINE Attributes](#) for information on the pipeline interval and other pipeline attributes.

STOP_PIPELINE Procedure

The procedure stops the data pipeline. When a pipeline is stopped, no future jobs are scheduled for the pipeline.

Syntax

```
DBMS_CLOUD_PIPELINE.STOP_PIPELINE(  
    pipeline_name      IN    VARCHAR2,  
    force              IN    BOOLEAN  DEFAULT FALSE  
);
```

Parameters

Parameter	Description
pipeline_name	Specifies a name for the pipeline. This parameter is mandatory.

Parameter	Description
force	If force parameter is passed as TRUE, then it will terminate any running jobs for the pipeline. Valid values: TRUE, FALSE Default value: FALSE

DBMS_CLOUD_PIPELINE Attributes

Attributes help to control and configure the behavior of a data pipeline.

Attributes

Note

As indicated in the **Pipeline Type** column, depending on the pipeline type LOAD or EXPORT, a pipeline supports a different set of attributes.

Attribute Name	Description	Pipeline Type	Modifiable After Pipeline Starts
credential_name	The name of the credential to access the source Cloud Object Storage. Default value: NULL. If you do not supply a credential_name, credential_name is set to NULL.	LOAD, EXPORT	Yes
field_list	Identifies the fields in the source files and their data types. This argument's syntax is the same as the field_list clause in regular Oracle external tables. For more information see <i>field_list</i> in Oracle Database 19c Utilities or Oracle Database 26ai Utilities . Default value: NULL The default value specifies the fields and their data types are determined by the columns in the table specified in table_name attribute.	LOAD	Yes
format	The options describing the format for the type of pipeline. - For a load pipeline, see DBMS_CLOUD Package Format Options. - For an export pipeline, see DBMS_CLOUD Package Format Options for EXPORT_DATA Datapump format is not supported for an export pipeline. This attribute is mandatory for both LOAD and EXPORT pipelines.	LOAD, EXPORT	Yes
interval	The time interval in minutes between consecutive executions of scheduled pipeline job. Default value: 15 minutes	LOAD, EXPORT	Yes

Attribute Name	Description	Pipeline Type	Modifiable After Pipeline Starts
key_column	A timestamp or date column in the specified table or query for exporting newer data continuously to object store. The last execution timestamp or date is tracked by Export pipeline and compared with the value in the key_column to identify new data for exporting to object store. Default value: NULL If key_column is not specified for an export pipeline, the entire contents of the table or query are uploaded to object store in each pipeline job execution.	EXPORT	No
location	This parameter specifies source file URI or directories and source files. You can use wildcards in the file names. This parameter is mandatory. Cloud source file URIs You can specify a source file URI for bucket or subfolder. You can use wildcards to specify subfolders or file name patterns. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. Example using wild cards: <pre>location_uri => 'https:// objectstorage.my\$region.oraclecloud.com/n/ namespace-string/b/bucketname/o/year=????/ month=??/*.csv'</pre> The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. Directory You can specify one directory and file name. The format to specify a directory is 'MY_DIR:filename.ext'. By default the directory name MY_DIR is a database object and is case-insensitive. The file name is case sensitive. You can only use wildcards to specify file names in a directory. The character "*" can be used as the wildcard for multiple characters, and the character "?" can be used as the wildcard for a single character. For example: 'MY_DIR: *' or 'MY_DIR: test?' Use double quotes to specify a case-sensitive directory name. For example: '"my_dir1":*, "my_dir2":Test?' To include a quote character, use two quotes. For example: 'MY_DIR: ' 'filename.ext'. This specifies the filename starts with a quote (').		

Attribute Name	Description	Pipeline Type	Modifiable After Pipeline Starts
priority	Specifies a string value that determines the number of parallel operations performed for the pipeline. - In a load pipeline, it determines the number of files loaded in parallel. - In an export pipeline, it determines the degree of parallelism for fetching data from the database. An operation with a higher priority consumes more database resources and is completed sooner. Valid values: - HIGH: Determines the number of parallel files handled using the database's ECPU count (OCPU count) if your database uses OCPUs). - MEDIUM: Determines the number of simultaneous processes using the concurrency limit for Medium service. The default value is 4. - LOW: Run the pipeline job in serial order. Default value: MEDIUM The maximum number of concurrent file operations is limited to 64.	LOAD, EXPORT	Yes
query	Specifies a SELECT statement so that only the required data is exported. The query determines the contents of the files you export as text files (CSV, JSON, Parquet, or XML) or dump files. For example: <pre>SELECT warehouse_id, quantity FROM inventories</pre> Default value: NULL For an export pipeline, either table_name or query is mandatory.	EXPORT	No
table_name	Specifies the name of the target table for loading or exporting data. For a load pipeline table_name is mandatory. For an export pipeline, either table_name or query is mandatory.	LOAD, EXPORT	No
table_owner	The name of the schema where the target table resides for loading or exporting data. Default value: NULL With a NULL value the target table is in the same schema as the user running the procedure.	LOAD, EXPORT	No

Summary of DBMS_EXT_TABLE_CACHE Subprograms

This table summarizes the subprograms included in the DBMS_EXT_TABLE_CACHE package.

Subprogram	Description
ADD_BY_LIKE Procedure	Adds one or more files that match the specified filters into an external table cache.
ADD_FILE Procedure	Adds a file into the external table cache.
ADD_TABLE Procedure	Adds files from the specified external table into an external table cache.
CLEAR Procedure	Clears an external table cache.
CREATE_CACHE Procedure	Creates an external table cache.
DISABLE Procedure	Disables an external table cache.
DROP_BY_LIKE Procedure	Drop files from an external table cache based on the specified filters.
DROP_CACHE Procedure	Drops an external table cache.
DROP_FILE Procedure	Removes the specified external table file from the external table cache.
ENABLE Procedure	Enables a previously disabled external table cache.
GET_USER_PROPERTY Function	Retrieves the caching preference for an external table cache.
RETIRE_FILES Procedure	Deletes one or more files from the cache that are older than the specified interval.
SET_USER_PROPERTY Procedure	Specifies the caching preference for an external table cache.
VALIDATE Procedure	Validates an external table cache.

ADD_BY_LIKE Procedure

The DBMS_EXT_TABLE_CACHE.ADD_BY_LIKE procedure loads one or more specified files from Object Store into the external table cache.

Syntax

```
DBMS_EXT_TABLE_CACHE.ADD_BY_LIKE (  
    owner          IN VARCHAR2,  
    table_name     IN VARCHAR2,  
    path_filters   IN CLOB,  
    esc_char       IN VARCHAR2 DEFAULT NULL,  
    force          IN BOOLEAN  DEFAULT FALSE);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.

Parameter	Description
path_filters	PATH_FILTERS is a JSON_ARRAY of path filters used to search for specified patterns in file URLs.
esc_char	Specifies the escape character to be used in the path filters. If the escape character precedes the '%' or '_' in the pattern, the special character is interpreted literally, and not as a special pattern-matching character. This parameter is optional and the default value for this parameter is NULL.
force	Forces the specified existing files to be overwritten in the cache even if the files were not modified. This parameter is optional and the default value for the force parameter is FALSE.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_BY_LIKE (
    owner      => 'SALES',
    table_name => 'STORE_SALES',
    path_filters => ['"https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata1.parquet",
                    "https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata2.parquet"]'
  );
END;
/
```

Usage Note

- By default, the DBMS_EXT_TABLE_CACHE.ADD_BY_LIKE procedure skips loading the files when:
 - The files were previously cached and still available in the cache.
 - The files have not been modified since they were last cached.

However, you can use the force parameter to overwrite the files in the cache even if the files were not modified.

ADD_FILE Procedure

The DBMS_EXT_TABLE_CACHE.ADD_FILE procedure loads the specified file into the external table cache.

Syntax

```
DBMS_EXT_TABLE_CACHE.ADD_FILE (
  owner      IN VARCHAR2,
  table_name IN VARCHAR2,
  file_url   IN VARCHAR2,
  force      IN BOOLEAN DEFAULT FALSE);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.
file_url	Specifies the file URL.
force	Forces the specified existing files to be overwritten in the cache even if the files were not modified. This parameter is optional and the default value for force parameter is FALSE.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_FILE (
    owner       => 'SALES',
    table_name  => 'STORE_SALES',
    file_url    => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata.parquet'
  );
END;
/
```

Usage Notes

- The DBMS_EXT_TABLE_CACHE.ADD_FILE procedure skips loading the file into the cache if the specified file exists in the cache and has not been modified since the file was last cached.
- You can use the force parameter to overwrite the file in the cache even if the file was not modified.

ADD_LATEST_FILES Procedure

The DBMS_EXT_TABLE_CACHE.ADD_LATEST_FILES loads one or more files into the external table cache. The files are loaded based on the time interval determined by the SINCE argument and the current timestamp.

Syntax

```
DBMS_EXT_TABLE_CACHE.ADD_LATEST_FILES (
  owner       IN  VARCHAR2,
  table_name  IN  VARCHAR2,
  since       IN  INTERVAL DAY TO SECOND,
  max_files   IN  NUMBER,
  force       IN  BOOLEAN DEFAULT FALSE);
```

Parameters

Parameter	Description
owner	Specifies the schema name.

Parameter	Description
table_name	Specifies the external table name.
since	The since parameter accepts an INTERVAL DAY TO SECOND value and is used to calculate the time interval between the since value and the current timestamp. The resultant time interval is then used to load the files, based on their last modified time.
max_files	Specify the max_files value to limit the number of files that can be loaded into the external table cache. This parameter is optional and defaults to unlimited when not specified.
force	Forces the specified existing files to be overwritten in the cache even if the files were not modified. This parameter is optional and the default value for this parameter is FALSE.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_LATEST_FILES (
    owner       => 'SALES',
    table_name  => 'STORE_SALES',
    since       => INTERVAL '7' DAY,
    max_files   => 5,
    force       => TRUE);
END;
/
```

Usage Note

- By default, the DBMS_EXT_TABLE_CACHE.ADD_LATEST_FILES procedure skips loading the files when:
 - The specified files were previously cached and still available in the cache.
 - The specified files have not been modified since they were last cached.However, you can use the force parameter to overwrite the files in the cache even if the files were not modified.

ADD_TABLE Procedure

The DBMS_EXT_TABLE_CACHE.ADD_TABLE procedure loads an entire table or a certain percentage of the external table into the cache.

Syntax

```
DBMS_EXT_TABLE_CACHE.ADD_TABLE (
  owner          IN VARCHAR2,
  table_name     IN VARCHAR2,
  percent_files  IN NUMBER DEFAULT NULL,
  force          IN BOOLEAN DEFAULT FALSE);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.
percent_files	Specifies the percentage of the table data to be cached. For example, 1 to 100. By default, all files are loaded.
force	Forces the specified files to be overwritten in the cache even if the files were not modified. This parameter is optional and the default value for force is FALSE.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.ADD_TABLE (
    owner      => 'SALES',
    table_name => 'STORE_SALES',
    percent_files => 50);
END;
/
```

Usage Notes

- The DBMS_EXT_TABLE_CACHE.ADD_TABLE attempts to cache the entire table or a specified percentage of the file content into the cache.
- The percent_files and force parameters are mutually exclusive.
- By default, the DBMS_EXT_TABLE_CACHE.ADD_TABLE procedure skips loading the files when:
 - The specified files were previously cached and still available in the cache.
 - The specified files have not been modified since they were last cached.

However, you can use the force parameter to overwrite the files in the cache even if the files were not modified.

CLEAR Procedure

The DBMS_EXT_TABLE_CACHE.CLEAR procedure removes all files from an external table cache while retaining the cache.

Syntax

```
DBMS_EXT_TABLE_CACHE.CLEAR (
  owner      IN VARCHAR2,
  table_name IN VARCHAR2);
```


Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.CLEAR (
    owner      => 'SALES',
    table_name => 'STORE_SALES');
END;
/
```

CREATE_CACHE Procedure

The `DBMS_EXT_TABLE_CACHE.CREATE_CACHE` procedure creates an external table cache in an Autonomous AI Database instance.

Syntax

```
DBMS_EXT_TABLE_CACHE.CREATE_CACHE (
  owner      IN  VARCHAR2,
  table_name IN  VARCHAR2,
  partition_type IN  VARCHAR2);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the name of the external table.
partition_type	Following are the valid values for the <code>partition_type</code> parameter: - FILE: partitions the external table cache by using the <code>FILE\$PATH</code> and <code>FILE\$NAME</code> invisible columns. - PATH: partitions the external table cache by using the <code>FILE\$PATH</code> invisible column. <code>file\$path</code>: Specifies the file path text up to the beginning of the object name. <code>file\$name</code>: Specifies the object name, including all the text that follows the final <code>'/'</code>. See External Table Metadata Columns for more information.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.CREATE_CACHE (
    owner      => 'SALES',
    table_name => 'STORE_SALES',
```

```
        partition_type =>  
'FILE');  
END;  
/
```

Usage Note

- `DBMS_EXT_TABLE_CACHE.CREATE_CACHE` creates the external table cache in an Autonomous AI Database instance. Creating a cache is similar to creating a table in the database schema.

DISABLE Procedure

The `DBMS_EXT_TABLE_CACHE.DISABLE` procedure disables the specified external table cache. The cache is flagged as disabled; however, the data within the cache is retained.

Syntax

```
DBMS_EXT_TABLE_CACHE.DISABLE (  
    owner          IN VARCHAR2,  
    table_name     IN VARCHAR2  
);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.

Example

```
BEGIN  
    DBMS_EXT_TABLE_CACHE.DISABLE (  
        owner          => 'SALES',  
        table_name     =>  
'STORE_SALES');  
  
END;  
/
```

Usage Note

- `DBMS_EXT_TABLE_CACHE.DISABLE` does not delete any data from the external table cache. Instead, it marks the cache as **DISABLED**, meaning the optimizer cannot use the cache for query rewrites.

DROP_BY_LIKE Procedure

The `DBMS_EXT_TABLE_CACHE.DROP_BY_LIKE` procedure drops one or more files from the external table cache. The files are dropped based on the specified filters.

Syntax

```
DBMS_EXT_TABLE_CACHE.DROP_BY_LIKE (  
    owner          IN VARCHAR2,  
    table_name     IN VARCHAR2,  
    path_filters   IN CLOB,  
    esc_char       IN VARCHAR2 DEFAULT NULL);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.
path_filters	<code>PATH_FILTERS</code> is a <code>JSON_ARRAY</code> of path filters used to search for specified patterns in file URLs.
esc_char	Specifies the escape character to be used in the path filters. If the escape character precedes the '%' or '_' in the pattern, the special character is interpreted literally, and not as a special pattern-matching character. This parameter is optional and the default value for this parameter is <code>NULL</code> .

Example

```
BEGIN  
    DBMS_EXT_TABLE_CACHE.DROP_BY_LIKE (  
        owner          => 'SALES',  
        table_name     => 'STORE_SALES',  
        path_filters   => '["https://swiftobjectstorage.us-  
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata1.parquet",  
                        "https://swiftobjectstorage.us-  
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata2.parquet"]'  
    );  
END;  
/
```

DROP_CACHE Procedure

The `DBMS_EXT_TABLE_CACHE.DROP_CACHE` procedure drops the specified external table cache. This procedure drops the cache and releases the storage space associated with the cache.

Syntax

```
DBMS_EXT_TABLE_CACHE.DROP_CACHE (  
    owner          IN   VARCHAR2,  
    table_name     IN   VARCHAR2  
);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.

Example

```
BEGIN
    DBMS_EXT_TABLE_CACHE.DROP_CACHE (
        owner      => 'SALES',
        table_name =>
'STORE_SALES');
END;
/
```

Usage Note

- Dropping a cache removes its metadata from the data dictionary and deletes all its cached data.

DROP_FILE Procedure

The `DBMS_EXT_TABLE_CACHE.DROP_FILE` procedure drops the specified file from an external table cache.

Syntax

```
DBMS_EXT_TABLE_CACHE.DROP_FILE (
    owner      IN VARCHAR2,
    table_name IN VARCHAR2,
    file_url   IN VARCHAR2);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.
file_url	Specifies the file URL.

Example

```
BEGIN
    DBMS_EXT_TABLE_CACHE.DROP_FILE (
        owner      => 'SALES',
        table_name => 'STORE_SALES',
        file_url   => 'https://swiftobjectstorage.us-
ashburn-1.oraclecloud.com/v1/your_namespace/your_bucket/salesdata.parquet'
    );
```

```
END;  
/
```

ENABLE Procedure

The `DBMS_EXT_TABLE_CACHE.ENABLE` procedure enables a previously disabled external table cache. When a cache is created, it is enabled by default.

Syntax

```
DBMS_EXT_TABLE_CACHE.ENABLE (  
    owner          IN    VARCHAR2,  
    table_name     IN    VARCHAR2  
);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.

Example

```
BEGIN  
    DBMS_EXT_TABLE_CACHE.ENABLE (  
        owner          => 'SALES',  
        table_name     => 'STORE_SALES'  
    );  
END;  
/
```

GET_USER_PROPERTY Function

The function returns the caching preference for the specified schema and returns a Number.

Syntax

```
DBMS_EXT_TABLE_CACHE.GET_USER_PROPERTY (  
    property_name  IN    VARCHAR2,  
    owner          IN    VARCHAR2 DEFAULT NULL);  
  
RETURN NUMBER;
```

Parameters

Parameter	Description
property_name	Specifies the property name. Following are the valid values for property_name: • MAX_CACHE_PERCENT • MAX_CACHE_SIZE
owner	Specifies the schema name.

Return Values

Return Value	Description
The MAX_CACHE_SIZE or MAX_CACHE_SIZE value in Number.	Depending on the property_name parameter the number returned is the MAX_CACHE_SIZE or MAX_CACHE_SIZE value.

Example

```
SET SERVEROUTPUT ON
DECLARE
  max_cache_sz NUMBER,
BEGIN
  max_cache_sz := DBMS_EXT_TABLE_CACHE.GET_USER_PROPERTY (
    property_name => 'MAX_CACHE_SIZE',
    owner        =>
'SALES');
END;
/
```

RETIRE_FILES Procedure

The `DBMS_EXT_TABLE_CACHE.RETIRE_FILES` drops files from the cache that are older than the specified interval. The files are deleted based on the time interval calculated using the `BEFORE` parameter value.

Syntax

```
DBMS_EXT_TABLE_CACHE.RETIRE_FILES (
  owner      IN VARCHAR2,
  table_name IN VARCHAR2,
  before     IN INTERVAL DAY TO SECOND
);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.

Parameter	Description
before	The before parameter accepts an INTERVAL DAY TO SECOND value and is used to calculate the time interval between the before value and the current timestamp. The resultant time interval is then used to delete the files from the cache.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.RETIRE_FILES (
    owner       => 'SALES',
    table_name  => 'STORE_SALES',
    before      => INTERVAL '30' DAY);
END;
/
```

SET_USER_PROPERTY Procedure

The DBMS_EXT_TABLE_CACHE.SET_USER_PROPERTY procedure sets the caching preference for a schema.

Syntax

```
DBMS_EXT_TABLE_CACHE.SET_USER_PROPERTY (
  property_name  IN   VARCHAR2,
  property_value IN   NUMBER,
  owner          IN   VARCHAR2 DEFAULT NULL);
```

Parameters

Parameter	Description
property_name	Specifies the property name. Following are the valid values for property_name: - MAX_CACHE_PERCENT - MAX_CACHE_SIZE
property_value	Specifies the property value.
owner	Specifies the schema name.

Example

```
BEGIN
  DBMS_EXT_TABLE_CACHE.SET_USER_PROPERTY (
    property_name => 'MAX_CACHE_PERCENT',
    property_value => 100,
    owner        =>
'SALES');
END;
/
```

VALIDATE Procedure

The `DBMS_EXT_TABLE_CACHE.VALIDATE` procedure validates the external table cache. An error is reported if the referenced external table is not found in the database.

Syntax

```
DBMS_EXT_TABLE_CACHE.VALIDATE (  
    owner          IN    VARCHAR2,  
    table_name     IN    VARCHAR2,  
    raise_errors   IN    BOOLEAN DEFAULT TRUE);
```

Parameters

Parameter	Description
owner	Specifies the schema name.
table_name	Specifies the external table name.
raise_errors	Reports when an external table cache is marked invalid.

Example

```
BEGIN  
    DBMS_EXT_TABLE_CACHE.VALIDATE (  
        owner          => 'SALES',  
        table_name     => 'STORE_SALES',  
        raise_errors   => TRUE);  
END;  
/
```

DBMS_PIPE Package

The `DBMS_PIPE` package lets two or more sessions in the same instance communicate.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure supports core `DBMS_PIPE` functionality as available in Oracle Database 19c, plus extensions.

See *DBMS_PIPE* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for details about the core `DBMS_PIPE` functionality provided in Oracle Database.

DBMS_PIPE Overview for Singleton Pipes

Pipe functionality has several potential applications: external service interface, debugging, independent transactions, and alerts.

On Autonomous AI Database the `DBMS_PIPE` package has extended functionality to support singleton pipes. See *DBMS_PIPE* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for more information.

Singleton pipe features in `DBMS_PIPE` provide the following:

- Ability to cache and retrieve a custom message, of up to 32,767 bytes, in Oracle database memory. The message size maximum of 32,767 bytes is applicable to all Pipes, including Singleton Pipes. Previous versions of DBMS_PIPE had a smaller maximum message size.
- Share the cached message across multiple database sessions with concurrent reads.
- Cache Invalidation methods:
 - Explicit cache invalidation controlled by user.
 - Cache invalidation after a user specified parameter (shelflife) time interval (in seconds).
- Declarative and easy to use PL/SQL APIs for caching.
- Supports both Read-Only and Read-Write databases.

A Singleton Pipe can be any one of the supported DBMS_PIPE types:

- **Implicit Pipe:** Automatically created when a message is sent with an unknown pipe name using the DBMS_PIPE.SEND_MESSAGE function.
- **Explicit Pipe:** Created using the DBMS_PIPE.CREATE_PIPE function with a user specified pipe name.
- **Public Pipe:** Accessible by any user with EXECUTE permission on DBMS_PIPE package
- **Private Pipe:** Accessible by sessions with the same user as the pipe creator.

Summary of DBMS_PIPE Subprograms for Singleton Pipes

This table lists the DBMS_PIPE subprograms and briefly describes them.

Table 7-2 DBMS_PIPE Package Subprograms

Subprogram	Description
CREATE_PIPE Function	Creates a pipe (necessary for private pipes)
NEXT_ITEM_TYPE Function • Oracle Database 19c PL/SQL Packages and Types Reference • Oracle Database 26ai PL/SQL Packages and Types Reference	Returns datatype of next item in buffer
PACK_MESSAGE Procedures • Oracle Database 19c PL/SQL Packages and Types Reference • Oracle Database 26ai PL/SQL Packages and Types Reference	Builds message in local buffer
PURGE Procedure • Oracle Database 19c PL/SQL Packages and Types Reference • Oracle Database 26ai PL/SQL Packages and Types Reference	Purges contents of named pipe
RECEIVE_MESSAGE Function	Copies message from named pipe into local buffer
RESET_BUFFER Procedure • Oracle Database 19c PL/SQL Packages and Types Reference • Oracle Database 26ai PL/SQL Packages and Types Reference	Purges contents of local buffer

Table 7-2 (Cont.) DBMS_PIPE Package Subprograms

Subprogram	Description
REMOVE_PIPE Function - Oracle Database 19c PL/SQL Packages and Types Reference Oracle Database 26ai PL/SQL Packages and Types Reference	Removes the named pipe
SEND_MESSAGE Function	Sends message on named pipe: This implicitly creates a public pipe if the named pipe does not exist
UNIQUE_SESSION_NAME Function - Oracle Database 19c PL/SQL Packages and Types Reference Oracle Database 26ai PL/SQL Packages and Types Reference	Returns unique session name
UNPACK_MESSAGE Procedures - Oracle Database 19c PL/SQL Packages and Types Reference Oracle Database 26ai PL/SQL Packages and Types Reference	Accesses next item in buffer

CREATE_PIPE Function

This function explicitly creates a public or private pipe. If the private flag is TRUE, then the pipe creator is assigned as the owner of the private pipe.

Explicitly-created pipes can only be removed by calling REMOVE_PIPE, or by shutting down the instance.

In order to create a Singleton Pipe, set the singleton parameter to TRUE. The following arguments are applicable to Singleton Pipes:

- singleton:** Indicates that the pipe should be created as a Singleton Pipe (default value: FALSE).
- shelflife:** Optionally specify a shelflife expiration (in seconds) of cached message in the Singleton Pipe. It can be used for implicit invalidation of message in Singleton Pipe.

The message shelflife in Singleton Pipe can also be specified when you send a message message (see [SEND_MESSAGE Function](#)).

Syntax

```
DBMS_PIPE.CREATE_PIPE (
    pipename      IN VARCHAR2,
    maxpipesize   IN INTEGER DEFAULT 66536,
    private       IN BOOLEAN DEFAULT TRUE,
    singleton     IN BOOLEAN DEFAULT FALSE,
    shelflife     IN INTEGER DEFAULT 0)
RETURN INTEGER;
```

Parameters

Table 7-3 CREATE_PIPE Function Parameters

Parameter	Description
pipename	Name of the pipe you are creating. You must use this name when you call <code>SEND_MESSAGE</code> and <code>RECEIVE_MESSAGE</code>. This name must be unique across the instance. Caution: Do not use pipe names beginning with <code>ORA\$</code>. These are reserved for use by procedures provided by Oracle. Pipename should not be longer than 128 bytes, and is case insensitive. At this time, the name cannot contain Globalization Support characters.
maxpipesize	The maximum size allowed for the pipe, in bytes. The total size of all of the messages on the pipe cannot exceed this amount. The message is blocked if it exceeds this maximum. The default maxpipesize is 66536 bytes. The maxpipesize for a pipe becomes a part of the characteristics of the pipe and persists for the life of the pipe. Callers of <code>SEND_MESSAGE</code> with larger values cause the maxpipesize to be increased. Callers with a smaller value use the existing, larger value. The default maxpipesize of 66536 is applicable for all Pipes.
private	Uses the default, <code>TRUE</code>, to create a private pipe. Public pipes can be implicitly created when you call <code>SEND_MESSAGE</code>.
singleton	Use <code>TRUE</code> to create a Singleton Pipe. Default value: <code>FALSE</code>
shelflife	Expiration time in seconds of a message cached in Singleton Pipe. After the specified shelflife time is exceeded, the message is no longer accessible from the Pipe. The parameter shelflife is only applicable to a Singleton Pipe. Default value is 0, implying that the message never expires.

Return Values

Table 7-4 CREATE_PIPE Function Return Values

Return	Description
0	Successful. If the pipe already exists and the user attempting to create it is authorized to use it, then Oracle returns 0, indicating success, and any data already in the pipe remains.
6	Failed to convert existing pipe to singleton pipe. - Implicit pipe with more than one existing message cannot be converted to a Singleton Pipe. - For an explicit pipe that is not Singleton, <code>DBMS_PIPE.SEND_MESSAGE</code> cannot send a message with singleton argument set to <code>TRUE</code>.
7	A non-zero value is given for the shelflife parameter and the pipe is not a singleton pipe.

Table 7-4 (Cont.) CREATE_PIPE Function Return Values

Return	Description
ORA-23322	Failure due to naming conflict. If a pipe with the same name exists and was created by a different user, then Oracle signals error ORA-23322, indicating the naming conflict.

Exceptions**Table 7-5 CREATE_PIPE Function Exception**

Exception	Description
Null pipe name	Permission error: Pipe with the same name already exists, and you are not allowed to use it.

Example

Create a Singleton Pipe with shelflife of 1 hour.

```
DECLARE
  l_status INTEGER;
BEGIN
  l_status := DBMS_PIPE.create_pipe(pipename => 'MY_PIPE1',
                                   private   => TRUE,
                                   singleton => TRUE,
                                   shelflife => 3600);
END;
/
```

RECEIVE_MESSAGE Function

This function copies the message into the local message buffer.

Syntax

```
DBMS_PIPE.RECEIVE_MESSAGE (
  pipename      IN VARCHAR2,
  timeout       IN INTEGER      DEFAULT maxwait,
  cache_func    IN VARCHAR2    DEFAULT NULL)
RETURN INTEGER;
```

Parameters**Table 7-6 RECEIVE_MESSAGE Function Parameters**

Parameter	Description
pipename	Name of the pipe on which you want to receive a message. Names beginning with ORA\$ are reserved for use by Oracle.

Table 7-6 (Cont.) RECEIVE_MESSAGE Function Parameters

Parameter	Description
timeout	Time to wait for a message, in seconds. A timeout of 0 lets you read without blocking. The timeout does not include the time spent in execution cache function specified in the cache_func parameter. Default value: is the constant MAXWAIT, which is defined as 86400000 (1000 days).
cache_func	Cache function name to automatically cache a message in a Singleton Pipe. The name of the function should be fully qualified with the owner schema: • <i>OWNER.FUNCTION_NAME</i> • <i>OWNER.PACKAGE.FUNCTION_NAME</i> Default value: NULL

Return Values**Table 7-7 RECEIVE_MESSAGE Function Return Values**

Return	Description
0	Success
1	Timed out. If the pipe was implicitly-created and is empty, then it is removed.
2	Record in the pipe is too large for the buffer.
3	An interrupt occurred.
8	Cache function can only be specified when using a Singleton Pipe.
ORA-23322	User has insufficient privileges to read from the pipe.

Usage Notes

To receive a message from a pipe, first call `RECEIVE_MESSAGE`. When you receive a message, it is removed from the pipe; hence, a message can only be received once. For implicitly-created pipes, the pipe is removed after the last record is removed from the pipe.

If the pipe that you specify when you call `RECEIVE_MESSAGE` does not already exist, then Oracle implicitly creates the pipe and waits to receive the message. If the message does not arrive within a designated timeout interval, then the call returns and the pipe is removed.

After receiving the message, you must make one or more calls to `UNPACK_MESSAGE` to access the individual items in the message. The `UNPACK_MESSAGE` procedure is overloaded to unpack items of type `DATE`, `NUMBER`, `VARCHAR2`, and there are two additional procedures to unpack `RAW` and `ROWID` items. If you do not know the type of data that you are attempting to unpack, then call `NEXT_ITEM_TYPE` to determine the type of the next item in the buffer.

Cache Function Parameter

Singleton Pipes support cache function to automatically cache a message in the pipe in case of the following two scenarios:

- Singleton Pipe is empty.

- Message in Singleton Pipe is invalid due to shelfLife time elapsed.

The name of the function should be fully qualified with the owner schema:

- `OWNER.FUNCTION_NAME`
- `OWNER.PACKAGE.FUNCTION_NAME`

To use a cache function the current session user that invokes `DBMS_PIPE.RECEIVE_MESSAGE` must have required privileges to execute the cache function.

Cache Function Syntax

```
CREATE OR REPLACE FUNCTION cache_function_name (  
    pipename IN VARCHAR2  
) RETURN INTEGER;
```

Parameter	Datatype	Description
pipename	VARCHAR2	Name of the Singleton Pipe.
Return	Description	
0	Success	
Non-zero	Failure value returned from <code>DBMS_PIPE.RECEIVE_MESSAGE</code>	

Define a cache function to provide encapsulation and abstraction of complexity from the reader sessions of Singleton Pipe. The typical operations within a cache function would be:

- Create a Singleton Pipe, for an Explicit Pipe, using `DBMS_PIPE.CREATE_PIPE`.
- Create the message to cache in the Singleton Pipe.
- Send Message to Singleton Pipe, optionally specifying a shelfLife for the implicit message.

Exceptions

Table 7-8 `RECEIVE_MESSAGE` Function Exceptions

Exception	Description
Null pipe name	Permission error. Insufficient privilege to remove the record from the pipe. The pipe is owned by someone else.

Example

```
DECLARE
    l_status INTEGER;
BEGIN
    l_status := DBMS_PIPE.receive_message(pipename => 'MY_PIPE1',
                                          timeout   => 1,
                                          cache_func =>
      'MY_USER.MY_CACHE_FUNC');
END;
/
```

SEND_MESSAGE Function

This function sends a message on the named pipe.

The message is contained in the local message buffer, which was filled with calls to `PACK_MESSAGE`. You can create a pipe explicitly using `CREATE_PIPE`, otherwise, it is created implicitly.

To create an implicit Singleton pipe, set the `singleton` parameter to `TRUE`. The following arguments are applicable to Singleton Pipes:

- `singleton`: Indicates that the pipe should be created as a Singleton Pipe (default value: `FALSE`).
- `shelflife`: Optionally specify a shelflife expiration of cached message in the Singleton Pipe. It can be used for implicit invalidation of message in Singleton Pipe. This argument is applicable for implicit as well as explicit Singleton pipes. A `shelflife` value specified in [SEND_MESSAGE Function](#) overwrites the `shelflife` specified for the Explicit Singleton Pipe in [CREATE_PIPE Function](#) and will be the default for any new messages cached in the Singleton Pipe.

Syntax

```
DBMS_PIPE.SEND_MESSAGE (  
    pipename      IN VARCHAR2,  
    timeout       IN INTEGER DEFAULT MAXWAIT,  
    maxpipesize   IN INTEGER DEFAULT 65536,  
    singleton     IN BOOLEAN DEFAULT FALSE,  
    shelflife     IN INTEGER DEFAULT 0)  
RETURN INTEGER;
```

Parameters

Table 7-9 SEND_MESSAGE Function Parameters

Parameter	Description
<code>pipename</code>	Name of the pipe on which you want to place the message. If you are using an explicit pipe, then this is the name that you specified when you called <code>CREATE_PIPE</code> . Caution: Do not use pipe names beginning with 'ORA\$'. These names are reserved for use by procedures provided by Oracle. <code>Pipename</code> should not be longer than 128 bytes, and is case-insensitive. At this time, the name cannot contain Globalization Support characters.
<code>timeout</code>	Time to wait while attempting to place a message on a pipe, in seconds. The default value is the constant <code>MAXWAIT</code> , which is defined as 86400000 (1000 days).

Table 7-9 (Cont.) SEND_MESSAGE Function Parameters

Parameter	Description
maxpipesize	Maximum size allowed for the pipe, in bytes. The total size of all the messages on the pipe cannot exceed this amount. The message is blocked if it exceeds this maximum. The default is 65536 bytes. The maxpipesize for a pipe becomes a part of the characteristics of the pipe and persists for the life of the pipe. Callers of SEND_MESSAGE with larger values cause the maxpipesize to be increased. Callers with a smaller value simply use the existing, larger value. Specifying maxpipesize as part of the SEND_MESSAGE procedure eliminates the need for a separate call to open the pipe. If you created the pipe explicitly, then you can use the optional maxpipesize parameter to override the creation pipe size specifications. The default maxpipesize of 65536 is applicable for all Pipes.
singleton	Use TRUE to create a Singleton Pipe. Default value: FALSE
shelflife	Expiration time in seconds of a message cached in Singleton Pipe. After the specified shelflife time is exceeded, the message is no longer accessible from the Pipe. The parameter shelflife is only applicable to a Singleton Pipe. Default value is 0, implying that the message never expires.

Return Values**Table 7-10 SEND_MESSAGE Function Return Values**

Return	Description
0	Success. If the pipe already exists and the user attempting to create it is authorized to use it, then Oracle returns 0, indicating success, and any data already in the pipe remains. If a user connected as SYSDBS/SYSOPER re-creates a pipe, then Oracle returns status 0, but the ownership of the pipe remains unchanged.
1	Timed out. This procedure can timeout either because it cannot get a lock on the pipe, or because the pipe remains too full to be used. If the pipe was implicitly-created and is empty, then it is removed.
3	An interrupt occurred. If the pipe was implicitly created and is empty, then it is removed.
6	Failed to convert existing pipe to singleton pipe. - Implicit pipe with more than one existing message cannot be converted to a Singleton Pipe. - For an explicit pipe that is not Singleton, DBMS_PIPE.SEND_MESSAGE cannot send a message with singleton argument set to TRUE.
7	A non-zero value is given for the shelflife parameter and the pipe is not a singleton pipe.

Table 7-10 (Cont.) SEND_MESSAGE Function Return Values

Return	Description
ORA-23322	Insufficient privileges. If a pipe with the same name exists and was created by a different user, then Oracle signals error ORA-23322, indicating the naming conflict.

Exceptions**Table 7-11 SEND_MESSAGE Function Exception**

Exception	Description
Null pipe name	Permission error. Insufficient privilege to write to the pipe. The pipe is private and owned by someone else.

DBMS_PIPE Package

The DBMS_PIPE package lets two or more sessions in the same instance communicate.

Oracle Autonomous AI Database on Dedicated Exadata Infrastructure supports core DBMS_PIPE functionality as available in Oracle Database 19c, plus extensions.

See *DBMS_PIPE* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for details about the core DBMS_PIPE functionality provided in Oracle Database.

DBMS_PIPE Overview for Persistent Messaging Pipes

Pipe functionality has several potential applications: external service interface, debugging, independent transactions, and alerts.

On Autonomous AI Database the DBMS_PIPE package has extended functionality to support persistent messaging pipes. See *DBMS_PIPE* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#) for more information.

Persistent Messages in DBMS_PIPE:

- Support the ability to send and retrieve very large messages.
- Support a large number of pipe messages.
- Support sharing of messages within a single database, across multiple databases, and across databases in different regions.
- Support multiple pipes using the same Cloud Object Store location URI.

Persistent messaging functionality allows two or more database sessions to communicate with messages that are stored in Cloud Object Store. Using this functionality messages in a pipe can be made available to only the current database or they can be made available to multiple databases in the same region or across different regions.

A Persistent Messaging Pipe can be any one of the supported DBMS_PIPE types:

- **Implicit Pipe:** Automatically created when a message is sent with an unknown pipe name using the DBMS_PIPE.SEND_MESSAGE function.

- **Explicit Pipe:** Created using the `DBMS_PIPE.CREATE_PIPE` function with a user specified pipe name.
- **Public Pipe:** Accessible by any user with `EXECUTE` permission on `DBMS_PIPE` package.
- **Private Pipe:** Accessible by sessions with the same user as the pipe creator.

Note

When send and receive messages across different databases using persistent messages, Oracle recommends you call `DBMS_PIPE.CREATE_PIPE` before sending or receiving messages. Creating an explicit pipe with `DBMS_PIPE.CREATE_PIPE` ensures that a pipe is created with the access permissions you want, either public or private (by setting the `PRIVATE` parameter to `FALSE` or using the default value `TRUE`).

DBMS_PIPE Limitation

The `DBMS_PIPE` package does not support sending messages between databases that use different character sets. For example, if you have one Autonomous AI Database instances that uses `AL32UTF8` and another instance that uses `WE8MSWIN1252`, you cannot send messages with `DBMS_PIPE` between these two databases. In this case, you the system will raise the error `ORA-12704` if you attempt to send messages with `DBMS_PIPE` between these two databases.

Summary of DBMS_PIPE Subprograms for Persistent Messaging

This table lists the `DBMS_PIPE` subprograms and briefly describes them.

Table 7-12 DBMS_PIPE Package Subprograms

Subprogram	Description
CREATE_PIPE Function	Creates a pipe (necessary for private pipes).
GET_CREDENTIAL_NAME Function	Returns the global <code>credential_name</code> variable value.
GET_LOCATION_URI Function	Returns the global <code>location_uri</code> variable value that is used as a default location URI for use when a message is stored in Cloud Object Store.
NEXT_ITEM_TYPE Function • Oracle Database 19c PL/SQL Packages and Types Reference • Oracle Database 26ai PL/SQL Packages and Types Reference	Returns datatype of next item in buffer.
PACK_MESSAGE Procedures • Oracle Database 19c PL/SQL Packages and Types Reference • Oracle Database 26ai PL/SQL Packages and Types Reference	Builds message in local buffer.
RECEIVE_MESSAGE Function	Copies message from named pipe into local buffer.
RESET_BUFFER Procedure • Oracle Database 19c PL/SQL Packages and Types Reference • Oracle Database 26ai PL/SQL Packages and Types Reference	Purges contents of local buffer.

Table 7-12 (Cont.) DBMS_PIPE Package Subprograms

Subprogram	Description
REMOVE_PIPE Function - Oracle Database 19c PL/SQL Packages and Types Reference Oracle Database 26ai PL/SQL Packages and Types Reference	Removes the named pipe.
SEND_MESSAGE Function	Sends message on a named pipe: This implicitly creates a public pipe if the named pipe does not exist.
SET_CREDENTIAL_NAME Procedure	Sets the <code>credential_name</code> variable that is used as a default credential for messages that are stored in Cloud Object Store.
SET_LOCATION_URI Procedure	Sets the global <code>location_uri</code> variable that is used as a default location URI for messages that are stored in Cloud Object Store.
UNIQUE_SESSION_NAME Function - Oracle Database 19c PL/SQL Packages and Types Reference Oracle Database 26ai PL/SQL Packages and Types Reference	Returns unique session name.
UNPACK_MESSAGE Procedures - Oracle Database 19c PL/SQL Packages and Types Reference Oracle Database 26ai PL/SQL Packages and Types Reference	Accesses next item in buffer.

CREATE_PIPE Function

This function explicitly creates a public or private pipe. If the `private` flag is `TRUE`, then the pipe creator is assigned as the owner of the private pipe.

Explicitly-created pipes can only be removed by calling `REMOVE_PIPE`, or by shutting down the instance.

Syntax

```
DBMS_PIPE.CREATE_PIPE (
    pipename      IN VARCHAR2,
    maxpipesize   IN INTEGER DEFAULT 66536,
    private       IN BOOLEAN DEFAULT TRUE)
RETURN INTEGER;
```

Parameters

Table 7-13 CREATE_PIPE Function Parameters

Parameter	Description
pipename	Name of the pipe you are creating. You must use this name when you call SEND_MESSAGE and RECEIVE_MESSAGE. This name must be unique across the instance. Caution: Do not use pipe names beginning with ORA\$. These are reserved for use by procedures provided by Oracle. Pipename should not be longer than 128 bytes, and is case insensitive. At this time, the name cannot contain Globalization Support characters.
maxpipesize	The maximum size allowed for the pipe, in bytes. The total size of all of the messages on the pipe cannot exceed this amount. The message is blocked if it exceeds this maximum. The default maxpipesize is 66536 bytes. The maxpipesize for a pipe becomes a part of the characteristics of the pipe and persists for the life of the pipe. Callers of SEND_MESSAGE with larger values cause the maxpipesize to be increased. Callers with a smaller value use the existing, larger value. The default maxpipesize of 65536 is applicable for all Pipes.
private	Uses the default, TRUE, to create a private pipe. Public pipes can be implicitly created when you call SEND_MESSAGE.

Return Values

Table 7-14 CREATE_PIPE Function Return Values

Return	Description
0	Successful. If the pipe already exists and the user attempting to create it is authorized to use it, then Oracle returns 0, indicating success, and any data already in the pipe remains.
ORA-23322	Failure due to naming conflict. If a pipe with the same name exists and was created by a different user, then Oracle signals error ORA-23322, indicating the naming conflict.

Exceptions

Table 7-15 CREATE_PIPE Function Exception

Exception	Description
Null pipe name	Permission error: Pipe with the same name already exists, and you are not allowed to use it.

Example

Create an explicit private named MY_PIPE1

```
DECLARE
    l_status INTEGER;
BEGIN
    l_status := DBMS_PIPE.create_pipe(
        pipename => 'MY_PIPE1',
        private  => TRUE);
END;
/
```

GET_CREDENTIAL_NAME Function

This function returns the global `credential_name` variable value for use when messages are stored to Cloud Object Store.

Syntax

```
DBMS_PIPE.GET_CREDENTIAL_NAME
    RETURN VARCHAR2;
```

Return Values

Return Value	Description
<code>credential_name</code>	The name of the credential to access the Cloud Object Storage.

Example

```
DECLARE
    credential_name    VARCHAR2(400)
BEGIN
    credential_name := DBMS_PIPE.GET_CREDENTIAL_NAME;
END;
/
```

GET_LOCATION_URI Function

This function returns the global `location_uri` variable value that can be used as a default location URI when pipe messages are stored to Cloud Object Store.

Syntax

```
DBMS_PIPE.GET_LOCATION_URI
    RETURN VARCHAR2;
```

Return Value

Return Value	Description
<code>location_uri</code>	The object URI.

Example

```
DECLARE
  location_uri      VARCHAR2(400)
BEGIN
  location_uri := DBMS_PIPE.GET_LOCATION_URI;
END;
/
```

RECEIVE_MESSAGE Function

This function copies the message into the local message buffer.

Syntax

```
DBMS_PIPE.RECEIVE_MESSAGE (
  pipename          IN VARCHAR2,
  timeout           IN INTEGER  DEFAULT maxwait,
  credential_name   IN VARCHAR2 DEFAULT null,
  location_uri      IN VARCHAR2)
RETURN INTEGER;
```

Parameters

Table 7-16 RECEIVE_MESSAGE Function Parameters

Parameter	Description
pipename	Name of the pipe on which you want to receive a message. Names beginning with ORA\$ are reserved for use by Oracle.
timeout	Time to wait for a message, in seconds. A timeout of 0 lets you read without blocking. The timeout does not include the time spent running the cache function specified with the cache_func parameter. Default value: is the constant MAXWAIT, which is defined as 86400000 (1000 days).
credential_name	The credential name for the cloud store used to store messages. The credential_name is a package argument that is by default initialized as NULL. You can set this value before calling DBMS_PIPE.RECEIVE_MESSAGE. The passed parameter value takes precedence over the global variable's value. The credential object must have EXECUTE and READ/WRITE privileges by the user running DBMS_PIPE.RECEIVE_MESSAGE.
location_uri	The location URI for the cloud store used to store messages. The location_uri is a global variable that by default is initialized as NULL. You can set this value before calling DBMS_PIPE.RECEIVE_MESSAGE. The passed parameter value takes precedence over the global variable's value.

Return Values

Table 7-17 RECEIVE_MESSAGE Function Return Values

Return	Description
0	Success
1	Timed out. If the pipe was implicitly-created and is empty, then it is removed.
2	Record in the pipe is too large for the buffer.
3	An interrupt occurred.
ORA-23322	User has insufficient privileges to read from the pipe.

Usage Notes

- To receive a message from a pipe, first call `RECEIVE_MESSAGE`. When you receive a message, it is removed from the pipe; hence, a message can only be received once. For implicitly-created pipes, the pipe is removed after the last record is removed from the pipe.
- If the pipe that you specify when you call `RECEIVE_MESSAGE` does not already exist, then Oracle implicitly creates the pipe and waits to receive the message. If the message does not arrive within a designated timeout interval, then the call returns and the pipe is removed.
- After receiving the message, you must make one or more calls to `UNPACK_MESSAGE` to access the individual items in the message. The `UNPACK_MESSAGE` procedure is overloaded to unpack items of type `DATE`, `NUMBER`, `VARCHAR2`, and there are two additional procedures to unpack `RAW` and `ROWID` items. If you do not know the type of data that you are attempting to unpack, then call `NEXT_ITEM_TYPE` to determine the type of the next item in the buffer.
- Persistent messages are guaranteed to either be written or read by exactly one process. This prevents message content inconsistency due to concurrent writes and reads. Using a persistent messaging pipe, `DBMS_PIPE` allows only one operation, sending a message or a receiving message to be active at a given time. However, if an operation is not possible due to an ongoing operation, the process retries periodically until the timeout value is reached.
- If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:
 - If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
 - If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

Exceptions

Table 7-18 RECEIVE_MESSAGE Function Exceptions

Exception	Description
Null pipe name	Permission error. Insufficient privilege to remove the record from the pipe. The pipe is owned by someone else.

SEND_MESSAGE Function

This function sends a message on the named pipe.

The message is contained in the local message buffer, which was filled with calls to `PACK_MESSAGE`. You can create a pipe explicitly using `CREATE_PIPE`, otherwise, it is created implicitly.

Syntax

```
DBMS_PIPE.SEND_MESSAGE (  
    pipename          IN VARCHAR2,  
    timeout           IN INTEGER DEFAULT MAXWAIT,  
    credential_name    IN VARCHAR2 DEFAULT null,  
    location_uri       IN VARCHAR2 )  
RETURN INTEGER;
```

Parameters

Table 7-19 SEND_MESSAGE Function Parameters

Parameter	Description
<code>credential_name</code>	The credential name for the cloud store used to store messages. The <code>credential_name</code> is a package argument that is by default initialized as NULL. You can set this value before calling <code>DBMS_PIPE.SEND_MESSAGE</code>. The passed parameter value takes precedence over the global variable's value. The credential object must have EXECUTE and READ/WRITE privileges by the user running <code>DBMS_PIPE.SEND_MESSAGE</code>.
<code>location_uri</code>	The location URI for the cloud store used to store messages. The <code>location_uri</code> is a global variable that by default is initialized as NULL. You can set this value before calling <code>DBMS_PIPE.SEND_MESSAGE</code>. The passed parameter value takes precedence over the global variable's value.
<code>maxpipesize</code>	Maximum size allowed for the pipe, in bytes. The total size of all the messages on the pipe cannot exceed this amount. The message is blocked if it exceeds this maximum. The default is 65536 bytes. The <code>maxpipesize</code> for a pipe becomes a part of the characteristics of the pipe and persists for the life of the pipe. Callers of <code>SEND_MESSAGE</code> with larger values cause the <code>maxpipesize</code> to be increased. Callers with a smaller value simply use the existing, larger value. Specifying <code>maxpipesize</code> as part of the <code>SEND_MESSAGE</code> procedure eliminates the need for a separate call to open the pipe. If you created the pipe explicitly, then you can use the optional <code>maxpipesize</code> parameter to override the creation pipe size specifications. The default <code>maxpipesize</code> of 65536 is applicable for all Pipes.

Table 7-19 (Cont.) SEND_MESSAGE Function Parameters

Parameter	Description
pipename	Name of the pipe on which you want to place the message. If you are using an explicit pipe, then this is the name that you specified when you called CREATE_PIPE. Caution: Do not use pipe names beginning with 'ORA\$'. These names are reserved for use by procedures provided by Oracle. Pipename should not be longer than 128 bytes, and is case-insensitive. At this time, the name cannot contain Globalization Support characters.
timeout	Time to wait while attempting to place a message on a pipe, in seconds. The default value is the constant MAXWAIT, which is defined as 86400000 (1000 days).

Return Values**Table 7-20 SEND_MESSAGE Function Return Values**

Return	Description
0	Success. If the pipe already exists and the user attempting to create it is authorized to use it, then Oracle returns 0, indicating success, and any data already in the pipe remains. If a user connected as SYSDBS/SYSOPER re-creates a pipe, then Oracle returns status 0, but the ownership of the pipe remains unchanged.
1	Timed out. This procedure can timeout either because it cannot get a lock on the pipe, or because the pipe remains too full to be used. If the pipe was implicitly-created and is empty, then it is removed.
3	An interrupt occurred. If the pipe was implicitly created and is empty, then it is removed.
ORA-23322	Insufficient privileges. If a pipe with the same name exists and was created by a different user, then Oracle signals error ORA-23322, indicating the naming conflict.

Usage Notes

- Persistent messages are guaranteed to either be written or read by exactly one process. This prevents message content inconsistency due to concurrent writes and reads. Using a persistent messaging pipe, DBMS_PIPE allows only one operation, sending a message or a receiving message to be active at a given time. However, if an operation is not possible due to an ongoing operation, the process retries periodically until the timeout value is reached.
- If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:
 - If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.

- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

Exceptions

Table 7-21 SEND_MESSAGE Function Exception

Exception	Description
Null pipe name	Permission error. Insufficient privilege to write to the pipe. The pipe is private and owned by someone else.

SET_CREDENTIAL_NAME Procedure

This procedure sets the `credential_name` variable that is used as a default credential when pipe messages are stored in Cloud Object Store.

Syntax

```
DBMS_PIPE.SET_CREDENTIAL_NAME (  
    credential_name  IN VARCHAR2 );
```

Parameters

Parameter	Description
<code>credential_name</code>	The name of the credential to access the Cloud Object Storage.

Usage Note

If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:

- If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

Example

```
BEGIN  
    DBMS_PIPE.SET_CREDENTIAL_NAME(  
        credential_name => 'my_cred1');  
END;  
/
```

SET_LOCATION_URI Procedure

This procedure sets the global `location_uri` variable.

Syntax

```
DBMS_PIPE.SET_LOCATION_URI (  
    location_uri  IN VARCHAR2 );
```

Parameter

Parameter	Description
location_uri	Object or file URI. The format of the URI depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.

Usage Note

If you use Oracle Cloud Infrastructure Object Storage to store messages, you can use Oracle Cloud Infrastructure Native URIs or Swift URIs. However, the location URI and the credential must match in type as follows:

- If you use a native URI format to access Oracle Cloud Infrastructure Object Storage, you must use Native Oracle Cloud Infrastructure Signing Keys authentication in the credential object.
- If you use Swift URI format to access Oracle Cloud Infrastructure Object Storage, you must use an auth token authentication in the credential object.

Example

```
BEGIN
  DBMS_PIPE.GET_LOCATION_URI(
    location_uri => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname1/');
END;
/
```

Autonomous AI Database Views

Data Catalog Views

Data Catalog integration with Autonomous AI Database provides numerous tables and views.

These tables and views help you understand:

- Available Data Catalog assets. Get information about any type of Data Catalog asset - including databases, object stores, and more.
- Information about the Data Catalog Object Storage assets and entities that have been synchronized with Autonomous AI Database. This includes details about how Data Catalog items (assets, folders and entities) map to Autonomous AI Database objects (i.e. schemas and external tables).
- Metadata sync executions. Review details about sync jobs, including any issues that may have occurred during synchronization.

View	Description
ALL_CLOUD_CATALOG_DATABASES View	Display information about OCI Data Catalog data assets and AWS Glue Data Catalog databases
ALL_CLOUD_CATALOG_TABLES View	Used to display information about data entities for OCI Data Catalogs and tables for AWS Glue Data Catalogs
ALL_DCAT_ASSETS View	List data catalog assets that this database is authorized to access

View	Description
ALL_DCAT_ATTRIBUTES View	List data catalog attributes this database is authorized to access
ALL_DCAT_CONNECTIONS View	A view that contains information about the data catalog(s) connected to this instance
ALL_DCAT_ENTITIES View	Lists logical entities this database is authorized to access
ALL_DCAT_FOLDERS View	List metadata for the Object Storage buckets containing the data files for the Logical Entities
ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View	List all accessible catalogs across all regions, along with the level of access privileges for each catalog
ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View	List all accessible catalogs in the current region, along with the level of access privileges for each catalog
ALL_GLUE_DATABASES View	Lists the AWS Glue Data Catalog databases that the data catalog credential is authorized to access
ALL_GLUE_TABLES View	Shows all AWS Glue Data Catalog tables that the data catalog credential is authorized to access
DCAT_ATTRIBUTES View	List the mapping of logical entity attributes to external table columns
DCAT_ENTITIES View	Describes the mapping of logical entities to external tables
DBMS_DCAT\$SYNC_LOG View	Provides easy access to the log table for the last sync operation executed by the current user

ALL_CLOUD_CATALOG_DATABASES View

Use view ALL_CLOUD_CATALOG_DATABASES to display information about OCI Data Catalog data assets and AWS Glue Data Catalog databases.

Column	Description
DCAT_CON_ID	CON1
CATALOG_ID	Data catalog unique identifier. OCI Data Catalog example: ocid1.datacatalog.oc1.ap-mumbai-1....y35a AWS Glue Data Catalog example: NULL 579294766787
NAME	Name of the data asset (OCI)/ database (AWS Glue). OCI Data Catalog example: OBJECT_STORE_AT_ASHBURN AWS Glue Data Catalog example: OBJECT_STORE_AT_N_CALIFORNIA
DESCRIPTION	Description of the data asset (OCI)/ database (AWS Glue). OCI Data Catalog example: Data stored in S3 (N. California) AWS Glue Data Catalog example: Data stored in S3 (N. California)

Column	Description
TIME_CREATED	The date and time the data asset (OCI) / databases (AWS Glue) were created in the data catalog. OCI Data Catalog example: 26-SEP-22 10.56.01.395000 PM +00:00 AWS Glue Data Catalog example: 2022-06-15T09:45:35+01:00
DETAILS	JSON document with metadata about each data entity (OCI) / database (AWS Glue). OCI Data Catalog example: <pre>{ "catalog-id": "ocid1.datacatalog.oc1.ap-mumbai-1.amaaa...", "description": null, "display-name": "OBJECT_STORE_AT_ASHBURN", "external-key": "https://swiftobjectstorage.us-ashburn-1...", "key": "bc95181c-3ac3-4959-9e5f-4e460d3fb82a", "lifecycle-state": "ACTIVE", "time-created": "2022-09-26T22:56:01.395000+00:00", "type-key": "3ea65bc5-f60d-477a-a591-f063665339f9", "uri": "/dcat/20190325/dataAssets/bc95181c-3ac3-4959-9e5f-4e460d3fb82a" }</pre> AWS Glue Data Catalog example: <pre>{ "Name": "dbmsdcatpoc", "Parameters": { "somekey": "somevalue" }, "CreateTime": "2022-06-15T09:45:35+01:00", "CreateTableDefaultPermissions": [{ "Principal": { "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS" }, "Permissions": ["ALL"] }], "CatalogId": "579294766787" }</pre>

ALL_CLOUD_CATALOG_TABLES View

View ALL_CLOUD_CATALOG_TABLES is used to display information about data entities for OCI Data Catalogs and tables for AWS Glue Data Catalogs.

Column	Description
DCAT_CON_ID	Unique identifier of the data catalog. The connection id. OCI Data Catalog example: CON1 AWS Glue Data Catalog example: CON1
CATALOG_ID	Data catalog unique identifier. OCI Data Catalog example: ocid1.datacatalog.oc1.ap-mumbai-1....y35a AWS Glue Data Catalog example: NULL 579294766787
DATABASE_NAME	Name of the data asset (OCI)/ database (AWS Glue). OCI Data Catalog example: OBJECT_STORE_AT_ASHBURN AWS Glue Data Catalog example: OBJECT_STORE_AT_N_CALIFORNIA
NAME	Name of the data entity (OCI) / table (AWS Glue). OCI Data Catalog example: BIKES_TRIPS AWS Glue Data Catalog example: BIKES_TRIPS
DESCRIPTION	Description of the data entity (OCI) / table (AWS Glue). OCI Data Catalog example: Table storing bike trips AWS Glue Data Catalog example: Table storing bike trips
TIME_CREATED	The date and time the data entity (OCI) / table (AWS Glue) was created in the data catalog. OCI Data Catalog example: 26-SEP-22 10.56.01.395000 PM +00:00 AWS Glue Data Catalog example: 2022-06-15T09:45:35+01:00
TIME_UPDATED	Last time a change was made to the data entity (OCI) / table (AWS Glue). OCI Data Catalog example: 26-SEP-22 10.56.01.395000 PM +00:00 AWS Glue Data Catalog example: 2022-06-15T09:45:35+01:00

Column	Description
DETAILS	JSON document with metadata about each each data entity (OCI) / table (AWS Glue) OCI Data Catalog example: <pre> { "business-name": null, "data-asset-key": "bc95181c-3ac3-4959-9e5f-...", "description": null, "display-name": "bikes_trips", "external-key": "LE: https://swiftobjectstorage.us-ashburn-1.oraclecloud.com/v1/..._trips", "folder-key": "9c4b542d-d6eb-4b83-bf59-...", "folder-name": "hive", "is-logical": true, "is-partition": false, "key": "fde30a69-a07c-478a-ab62-...", "lifecycle-state": "ACTIVE", "object-storage-url": "https://objectstorage.us-ashburn-1.oraclecloud.com/n/...", "path": "OBJECT_STORE_AT_ASHBURN/hive/hive", "pattern-key": "db21b3f1-1508-4045-aa80-...", "properties": { "default": { "CONTENT-LENGTH": "4310321", "LAST-MODIFIED": "Fri, 9 Oct 2020 20:16:52 UTC", "archivedPECount": "0", "dataEntityExpression": "{logicalEntity:[^/]+}.db/{logicalEntity:[^/]+}/.*", "harvestedFile": "bikes.db/trips/p_start_month=2019-09/000000_0", "patternName": "bikes_trips" }, "harvestProps": { "characterSet": "UTF8", "compression": "none", "type": "PARQUET" } }, "realized-expression": "bikes.db/trips/.*", "time-created": "2022-09-26T22:56:35.063000+00:00", "time-updated": "2022-09-26T22:56:35.063000+00:00", "type-key": "6753c3af-7f88-44b9-be52-1d57bef462fb", "updated-by-id": "ocid1.user.oc1..r5l3tov7a", "uri": "/dcat/20190325/dataAssets/bc95181c-3ac3-4959-9e5f-..." } </pre> AWS Glue Data Catalog example: <pre> { "Name": "bikes_trips", </pre>

Column	Description
	<pre> "DatabaseName": "dbmsdcatpoc", "Owner": "owner", "CreateTime": "2022-06-23T13:24:20+01:00", "UpdateTime": "2022-06-23T13:24:20+01:00", "LastAccessTime": "2022-06-23T13:24:20+01:00", "Retention": 0, "StorageDescriptor": { "Columns": [{ "Name": "trip_duration", "Type": "int" }, { "Name": "start_month", "Type": "string" }, ...], "Location": "s3://dbmsdcatpoc/hive/bikes.db/trips/", "InputFormat": "org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputForma t", "OutputFormat": "org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputForm at", "Compressed": false, "NumberOfBuckets": -1, "SerdeInfo": { "SerializationLibrary": "org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe" , "Parameters": { "serialization.format": "1" } }, "BucketColumns": [], "SortColumns": [], "Parameters": { "CrawlerSchemaDeserializerVersion": "1.0", "CrawlerSchemaSerializerVersion": "1.0", "UPDATED_BY_CRAWLER": "crawler-bikes", "averageRecordSize": "86", "classification": "parquet", "compressionType": "none", "objectCount": "12", "recordCount": "404947", "sizeKey": "35312159", "typeOfData": "file" }, "StoredAsSubDirectories": false }, "PartitionKeys": [</pre>

Column	Description
	<pre> { "Name": "p_start_month", "Type": "string" }], "TableType": "EXTERNAL_TABLE", "Parameters": { "CrawlerSchemaDeserializerVersion": "1.0", "CrawlerSchemaSerializerVersion": "1.0", "UPDATED_BY_CRAWLER": "crawler-bikes", "averageRecordSize": "86", "classification": "parquet", "compressionType": "none", "objectCount": "12", "recordCount": "404947", "sizeKey": "35312159", "typeOfData": "file" }, "CreatedBy": "arn:aws:sts::579294766787:assumed-role/ AWSGlueServiceRole-dbmsdcatt/AWS-Crawler", "IsRegisteredWithLakeFormation": false, "CatalogId": "579294766787", "VersionId": "0" } </pre>

Example

ALL_DCAT_ASSETS View

The Data Catalog assets that this database is authorized to access.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2 (4000)	Connection identifier that is unique within the instance
KEY	VARCHAR2(4000)	Asset key
DISPLAY_NAME	VARCHAR2(4000)	Asset display name
DESCRIPTION	VARCHAR2(4000)	Asset description
CATALOG_ID	VARCHAR2(4000)	OCID for the Data Catalog containing the asset
EXTERNAL_KEY	VARCHAR2(4000)	Base Object Storage URI for the asset
URI	VARCHAR2(4000)	Asset URI for the Data Catalog API
TIME_CREATED	TIMESTAMP(6) WITH TIMEZONE	The date and time the data asset was created
TYPE_KEY	VARCHAR2(4000)	The key of the data asset type (currently, only Object Storage data assets are supported). Type keys can be found via the '/' types' Data Catalog endpoint.

Column	Datatype	Description
LIFECYCLE_STATE	VARCHAR2(4000)	The current state of the data asset. For more information on possible life cycle states, see the Data Catalog DataAsset Reference for a list of possible states for lifecycleState.

ALL_DCAT_ATTRIBUTES View

The Data Catalog attributes this database is authorized to access.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2 (4000)	Connection identifier that is unique within the instance
KEY	NUMBER	Attribute key
DISPLAY_NAME	VARCHAR2(4000)	Attribute display name
BUSINESS_NAME	VARCHAR2(4000)	Attribute business name
DESCRIPTION	VARCHAR2(4000)	Attribute description
DATA_ASSET_KEY	VARCHAR2(4000)	Data asset key
FOLDER_KEY	VARCHAR2(4000)	Folder key
ENTITY_KEY	VARCHAR2(4000)	Entity key
EXTERNAL_KEY	VARCHAR2(4000)	Unique external key for the attribute
LENGTH	NUMBER	Maximum allowed length of the attribute value
PRECISION	NUMBER	Precision of the attribute value (usually applies to float data type)
SCALE	NUMBER	Scale of the attribute value (usually applies to float data type)
IS_NULLABLE	NUMBER	Identifies if this attribute can be assigned null values
URI	VARCHAR2(4000)	URI to the attribute instance in the Data Catalog API
LIFECYCLE_STATE	VARCHAR2(4000)	The current state of the attribute. For more information on possible life cycle states, see the Data Catalog Attribute Reference for a list of possible states for lifecycleState.
TIME_CREATED	TIMESTAMP(6) WITH TIME ZONE	The date and time the attribute was created
EXTERNAL_DATA_TYPE	VARCHAR2(4000)	Data type of the attribute as defined in the external system
MIN_COLLECTION_COUNT	NUMBER	Minimum number of elements, if the type of the attribute is a collection type
MAX_COLLECTION_COUNT	NUMBER	Maximum number of elements, if the type of the attribute is a collection type
DATATYPE_ENTITY_KEY	VARCHAR2(4000)	Entity key that represents the datatype of this attribute, applicable if this attribute is a complex type
EXTERNAL_DATATYPE_ENTITY_KEY	VARCHAR2(4000)	External entity key that represents the datatype of this attribute, applicable if this attribute is a complex type

Column	Datatype	Description
PARENT_ATTRIBUTE_KEY	VARCHAR2(4000)	Attribute key that represents the parent attribute of this attribute, applicable if the parent attribute is of complex datatype
EXTERNAL_PARENT_ATTRIBUTE_KEY	VARCHAR2(4000)	External attribute key that represents the parent attribute of this attribute, applicable if the parent attribute is of complex type
PATH	VARCHAR2(4000)	Full path of the attribute

ALL_DCAT_CONNECTIONS View

A view that contains information about the data catalog(s) connected to this instance.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2(4000)	Connection identifier that is unique within the instance
COMPARTMENT_ID	VARCHAR2(4000)	OCID for the compartment where the Data Catalog instance resides
INSTANCE_ID	VARCHAR2(4000)	OCID for the Data Catalog instance
REGION	VARCHAR2(4000)	Region for the Data Catalog instance
ENDPOINT	VARCHAR2(4000)	Endpoint for the Data Catalog instance
CREATED	TIMESTAMP	When the Data Catalog instance was created
NAME	VARCHAR2(4000)	Name of the Data Catalog instance
LAST_UPDATED	TIMESTAMP	Timestamp of the last update of the connection to the Data Catalog instance
LATEST_OPERATION_ID	NUMBER	The id of the last synchronization operation
DATA_CATALOG_CREDENTIAL	VARCHAR2(128)	Credential used for accessing the Data Catalog
OBJECT_STORE_CREDENTIAL	VARCHAR2(128)	Credential used by the external table driver for accessing the Object Store

ALL_DCAT_ENTITIES View

The Data Catalog logical entities this database is authorized to access.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2(4000)	Connection identifier that is unique within the instance
CATALOG_ID	VARCHAR2(4000)	OCID for the Data Catalog containing the asset
KEY	VARCHAR2(4000)	Entity key
DISPLAY_NAME	VARCHAR2(4000)	Entity display name
BUSINESS_NAME	VARCHAR2(4000)	Entity business name
DESCRIPTION	VARCHAR2(4000)	Logical entity description
DATA_ASSET_KEY	VARCHAR2(4000)	Asset key
FOLDER_KEY	VARCHAR2(4000)	Folder unique key
FOLDER_NAME	VARCHAR2(4000)	Folder name (bucket)

Column	Datatype	Description
EXTERNAL_KEY	VARCHAR2(4000)	External key for the logical entity
PATTERN_KEY	VARCHAR2(4000)	Key of the associated pattern for the logical entity
REALIZED_EXPRESSION	VARCHAR2(4000)	The regular expression used to obtain the files for this logical entity
PATH	VARCHAR2(4000)	Full path for the logical entity
TIME_CREATED	TIMESTAMP(6) WITH TIME ZONE	Date and time the entity was created
TIME_UPDATED	TIMESTAMP(6) WITH TIME ZONE	Last time a change was made to the data entity
UPDATED_BY_ID	VARCHAR2(4000)	OCID of the user who updated this object in the Data Catalog
URI	VARCHAR2(4000)	URI of the entity instance in the API
LIFECYCLE_STATE	VARCHAR2(4000)	The current state of the entity. For more information on possible life cycle states, see the Data Catalog Entity Reference for a list of possible states for lifecycleState.

ALL_DCAT_FOLDERS View

Metadata for the Object Storage buckets containing the data files for the Logical Entities.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2(4000)	Connection identifier that is unique within the instance
CATALOG_ID	VARCHAR2(4000)	OCID for the Data Catalog containing the asset
KEY	VARCHAR2(4000)	Folder key
DISPLAY_NAME	VARCHAR2(4000)	Folder display name
BUSINESS_NAME	VARCHAR2(4000)	Folder business name
DESCRIPTION	VARCHAR2(4000)	Folder description
DATA_ASSET_KEY	VARCHAR2(4000)	Key for the data asset containing the folder
PARENT_FOLDER_KEY	VARCHAR2(4000)	Key for the parent folder (currently, this is the data asset key)
PATH	VARCHAR2(4000)	Full path for the folder
EXTERNAL_KEY	VARCHAR2(4000)	Object Storage URI for the bucket
TIME_EXTERNAL	TIMESTAMP(6) WITH TIMEZONE	The last modified timestamp of this folder
TIME_CREATED	TIMESTAMP(6) WITH TIMEZONE	The date/time the folder was created
URI	VARCHAR2(4000)	URI to the folder instance in the Data Catalog API.
LIFECYCLE_STATE	VARCHAR2(4000)	The current state of the folder. For more information on possible life cycle states, see the Data Catalog Folder Reference for a list of possible states for lifecycleState.

ALL_DCAT_GLOBAL_ACCESSIBLE_CATALOGS View

This view lists all accessible catalogs across all regions, along with the level of access privileges for each catalog.

Column	Datatype	Description
CATALOG_ID	VARCHAR2(4000)	Catalog OCID
CATALOG_NAME	VARCHAR2(4000)	Name of the catalog
CATALOG_REGION	VARCHAR2(4000)	Name of the catalog region
CATALOG_SCORE	NUMBER	The catalog score is a numeric value calculated from the privileges configured for the Data Catalog access credential. A higher catalog score means greater privileges, which may equate to a higher likelihood that this catalog is intended for use with this Autonomous AI Database instance.

ALL_DCAT_LOCAL_ACCESSIBLE_CATALOGS View

This view lists all accessible catalogs in the current region, along with the level of access privileges for each catalog.

Column	Datatype	Description
CATALOG_ID	VARCHAR2(4000)	Catalog OCID
CATALOG_NAME	VARCHAR2(4000)	Name of the catalog
CATALOG_SCORE	NUMBER	The catalog score is a numeric value calculated from the privileges configured for the Data Catalog access credential. A higher catalog score means greater privileges, which may equate to a higher likelihood that this catalog is intended for use with this Autonomous AI Database instance.

ALL_GLUE_DATABASES View

The AWS Glue Data Catalog databases that the data catalog credential is authorized to access.

Column	Data Type	Description
DCAT_CON_ID	VARCHAR2(4000)	Unique identifier of data catalog connection id.
CATALOG_ID	VARCHAR2(255)	Data Catalog unique identifier.
NAME	VARCHAR2(255)	Name of the database.
DESCRIPTION	VARCHAR2(2048)	Description of the database.
LOCATION_URI	VARCHAR2(1024)	The location of the database.
CREATE_TIME	TIMESTAMP	The time that the database was created in the data catalog.

Column	Data Type	Description
PARAMETERS	CLOB	JSON document with key-value pairs that define parameters and properties of the database.
TARGET_DATABASE	VARCHAR2(4000)	JSON document that describes a target database for resource linking in AWS.

ALL_GLUE_TABLES View

This view shows all AWS Glue Data Catalog tables that the data catalog credential is authorized to access.

Column	Data Type	Description
DCAT_CON_ID	VARCHAR2(4000)	Unique identifier of data catalog connection id.
CATALOG_ID	VARCHAR2(255)	Catalog identifier
DATABASE_NAME	VARCHAR2(255)	Database name
NAME	VARCHAR2(255)	Table name
TABLE_TYPE	VARCHAR2(255)	Table type
CLASSIFICATION	VARCHAR2(255)	
DESCRIPTION	VARCHAR2(2048)	Table description
OWNER	VARCHAR2(255)	Table owner
CREATED_BY	VARCHAR2(255)	Table creator
CREATE_TIME	TIMESTAMP	The time the table was created in the data catalog.
LAST_ANALYZED_TIME	TIMESTAMP	The last time column statistics were computed for this table.
LAST_ACCESS_TIME	TIMESTAMP	The last time the table was accessed.
UPDATE_TIME	TIMESTAMP	The last time the table was updated.
IS_REGISTERED_WITH_LAKE_FORMATI ON	NUMBER	Indicates whether the table is registered with AWS lake formation.
PARAMETERS	CLOB	JSON document with key-value pairs that define properties of the table.
PARTITION_KEYS	CLOB	JSON document with a list of columns by which the table is partitioned.
RETENTION	NUMBER	The retention time for this table.
STORAGE_DESCRIPTION	CLOB	JSON document with information about the physical storage of a table.
TARGET_TABLE	VARCHAR2(4000)	JSON document describing a target table used for resource linking in AWS.
VERSION_ID	VARCHAR2(255)	The version identifier for the table.
VIEW_EXPANDED_TEXT	CLOB	Introduced by AWS Glue for compatibility with Hive. Not used by AWS Glue.
VIEW_ORIGINAL_TEXT	CLOB	Introduced by AWS Glue for compatibility with Hive. Not used by AWS Glue.

DCAT_ATTRIBUTES View

Lists the mapping of logical entity attributes to external table columns.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2 (4000)	Connection identifier that is unique within the instance
ASSET_KEY	VARCHAR2(4000)	Data Catalog asset key
ENTITY_KEY	VARCHAR2(4000)	Data Catalog entity key
ATTRIBUTE_KEY	VARCHAR2(4000)	Data Catalog attribute key
ORACLE_COLUMN_NAME	VARCHAR2(128)	Mapped column name

DCAT_ENTITIES View

Describes the mapping of logical entities to external tables.

Column	Datatype	Description
DCAT_CON_ID	VARCHAR2(4000)	Connection identifier that is unique within the instance
ASSET_KEY	VARCHAR2(4000)	Data Catalog asset key
ENTITY_KEY	VARCHAR2(4000)	Data Catalog entity key
FOLDER_KEY	VARCHAR2(4000)	Data Catalog folder key
ORACLE_TABLE_NAME	VARCHAR2(128)	Mapped table name
ORACLE_SCHEMA_NAME	VARCHAR2(128)	Mapped schema name
ENTITY_ORACLE_DB_SCHEMA	VARCHAR2(4000)	The entity's oracle-db-schema custom property used to derive the schema
ASSET_ORACLE_DB_SCHEMA	VARCHAR2(4000)	The data asset's oracle-db-schema custom property used to derive the schema
FOLDER_ORACLE_DB_SCHEMA	VARCHAR2(4000)	The folder's oracle-db-schema custom property used to derive the schema

DBMS_DCAT\$SYNC_LOG View

The DBMS_DCAT\$SYNC_LOG view provides easy access to the log table for the last sync operation executed by the current user.

Every call to the RUN_SYNC procedure is logged to a new log table, pointed to by the LOGFILE_TABLE field of USER_LOAD_OPERATIONS. The log tables are automatically dropped after 2 days, and users can clear all sync logs using the DELETE_ALL_OPERATIONS Procedure where type is DCAT_SYNC.

The DBMS_DCAT\$SYNC_LOG view automatically identifies the latest log table. The schema for the DBMS_DCAT\$SYNC_LOG view is described below and the access permissions are identical to those of the individual log tables. By default READ is granted to the dbms_dcat role and to the ADMIN user.

The log tables have the following format:

Column	Datatype	Description
LOG_TIMESTAMP	TIMESTAMP	Timestamp for the log entry.
LOG_LEVEL	VARCHAR2(32)	The entry log level can have one of the following values: OFF, FATAL, ERROR, WARN, INFO, DEBUG, TRACE, ALL.
LOG_DETAILS	VARCHAR2(32767)	The log message.

DBMS_CLOUD_AI Views

The DBMS_CLOUD_AI package uses the following views.

DBA_CLOUD_AI_PROFILES View

The list of profiles created in the current database can be obtained by querying DBA_CLOUD_AI_PROFILES. The view displays the details of AI profiles in the database.

Only the ADMIN user is privileged to access DBA_CLOUD_AI_PROFILES. You cannot grant access to other users.

Column	Datatype	NULL	Description
PROFILE_ID	NUMBER	NOT NULL	Unique number assigned to the AI profile
PROFILE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the AI profile
STATUS	VARCHAR2(7)	NOT NULL	Current status of AI profile - ENABLED or DISABLED
DESCRIPTION	VARCHAR2(4000)	NOT NULL	User specified description for the AI profile
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation time for the AI profile.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the AI profile

USER_CLOUD_AI_PROFILES View

The list of profiles created in the current database can be obtained by querying USER_CLOUD_AI_PROFILES. The view displays the details of AI profiles in your schema.

Only the user is privileged to access USER_CLOUD_AI_PROFILES. You cannot grant access to other users.

Column	Datatype	NULL	Description
PROFILE_ID	NUMBER	NOT NULL	Unique number assigned to the AI profile
PROFILE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile
STATUS	VARCHAR2(7)	NOT NULL	Current status of AI profile - ENABLED or DISABLED
DESCRIPTION	VARCHAR2(4000)	-	User specified description for the AI profile
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation time for the AI profile

Column	Datatype	NULL	Description
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the AI profile

DBA_CLOUD_AI_PROFILE_ATTRIBUTES View

Attributes of AI profile manage the configuration and behavior of the AI profile. The view displays details of all AI profiles in the database.

Only the ADMIN user is privileged to access DBA_CLOUD_AI_PROFILE_ATTRIBUTES. You cannot grant access to other users.

Column	Datatype	NULL	Description
PROFILE_ID	NUMBER	NOT NULL	Unique number assigned to the AI profile
PROFILE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the AI profile
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile attribute
ATTRIBUTE_VALUE	CLOB	-	Value of the AI profile attribute
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time of the AI profile attribute

USER_CLOUD_AI_PROFILE_ATTRIBUTES View

Attributes of AI profile manage the configuration and behavior of the AI profile. The view displays details of AI profiles in your schema.

Only the user is privileged to access USER_CLOUD_AI_PROFILE_ATTRIBUTES. You cannot grant access to other users.

Column	Datatype	NULL	Description
PROFILE_ID	NUMBER	NOT NULL	Unique number assigned to the AI profile
PROFILE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the AI profile attribute
ATTRIBUTE_VALUE	CLOB	-	Value of the AI profile attribute
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time of the AI profile attribute

DBA_CLOUD_VECTOR_INDEXES View

You can obtain a list of vector indexes created in the current database by querying DBA_CLOUD_VECTOR_INDEXES. The view displays the details of all available vector indexes in the entire database.

Only the administrator is privileged to access DBA_CLOUD_VECTOR_INDEXES. You cannot grant access to other users.

Column	Datatype	NULL	Description
INDEX_ID	NUMBER	NOT NULL	Unique number assigned to the index
INDEX_NAME	VARCHAR2(128)	NOT NULL	Name of the index

Column	Datatype	NULL	Description
STATUS	VARCHAR2(7)	-	Current status of index: ENABLED or DISABLED
DESCRIPTION	CLOB	-	User specified description for the index
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation date and time of the index
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification date and time of the index

USER_CLOUD_VECTOR_INDEXES View

You can obtain a list of vector indexes you created in the current database by querying `USER_CLOUD_VECTOR_INDEXES`. The view displays the details of all vector indexes that you created in your schema or a PDB.

You have the privilege to access `USER_CLOUD_VECTOR_INDEXES`. You cannot grant access to other users.

Column	Datatype	NULL	Description
INDEX_ID	NUMBER	NOT NULL	Unique number assigned to the index
INDEX_NAME	VARCHAR2(128)	NOT NULL	Name of the index
STATUS	VARCHAR2(7)	-	Current status of index: ENABLED or DISABLED
DESCRIPTION	VARCHAR2(4000)	-	User specified description for the index
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation date and time of the index
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification date and time of the index

DBA_CLOUD_VECTOR_INDEX_ATTRIBUTES View

Query the `DBA_CLOUD_VECTOR_INDEX_ATTRIBUTES` view to display details about the attributes of vector indexes across the entire database. Attributes of the view manage the configuration and behavior of the vector index.

Only the administrator is privileged to access `DBA_CLOUD_VECTOR_INDEX_ATTRIBUTES`. You cannot grant access to other users.

Column	Datatype	NULL	Description
INDEX_ID	NUMBER	NOT NULL	Unique number assigned to the vector index
INDEX_NAME	VARCHAR2(128)	NOT NULL	Name of the vector index
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the vector index
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the vector index attribute
ATTRIBUTE_VALUE	CLOB	-	Value for the vector index attribute_name
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification date and time of the vector index

USER_CLOUD_VECTOR_INDEX_ATTRIBUTES View

Query the USER_CLOUD_VECTOR_INDEX_ATTRIBUTES view to display details about the attributes of vector indexes that you created in your schema or PDB. Attributes of the view manage the configuration and behavior of the vector index.

You have the privilege to access USER_CLOUD_VECTOR_INDEX_ATTRIBUTES. You cannot grant access to other users.

Column	Datatype	NULL	Description
INDEX_ID	NUMBER	NOT NULL	Unique number assigned to the vector index
INDEX_NAME	VARCHAR2(128)	NOT NULL	Name of the vector index
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the vector index attribute
ATTRIBUTE_VALUE	CLOB	-	Value for the vector index attribute_name
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification date and time of the vector index

USER_CLOUD_AI_CONVERSATIONS View

The view displays details on conversations in your schema.

The current user can see the conversations they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
CONVERSATION_ID	VARCHAR2(36)	NOT NULL	Stores the unique number assigned to a conversation.
CONVERSATION_TITLE	VARCHAR2(128)	NOT NULL	Specifies the title for the conversation.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the conversation. This can be NULL if the conversations is not used.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the conversation was created.
MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the conversation.
RETENTION_DAYS	NUMBER	-	Specifies the number of days to retain the conversation history.
CONVERSATION_LENGTH	NUMBER	-	Number of recent prompts and responses from the conversation that are sent to the LLM along with the current prompt.

USER_CLOUD_AI_CONVERSATION_PROMPTS View

The view displays details of the prompts in a conversation in your schema.

The current user can see the conversation prompts they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
CONVERSATION_PROMPT_ID	VARCHAR2(36)	NOT NULL	Stores the unique number assigned to a prompt.
CONVERSATION_ID	VARCHAR2(36)	NOT NULL	Stores the unique number assigned to a conversation.
CONVERSATION_TITLE	VARCHAR2(128)	NOT NULL	Specifies the name for the conversation.
PROFILE_NAME	VARCHAR2(128)	-	Specifies the AI profile used for the conversation.
PROMPT_ACTION	VARCHAR2(11)	NOT NULL	Specifies the type of action performed for the prompt (For example, narrate, runsql, chat).
PROMPT	CLOB	-	Represents a user-provided natural language input.
PROMPT_RESPONSE	CLOB	-	The LLM's response generated for the user prompt.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the prompt was created.
MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the prompt was modified.
CLIENT_IDENTIFIER	VARCHAR2(128)	-	Client identifier of the session where the prompt is created.
CLIENT_IP	VARCHAR2(128)	-	Unique identifier assigned to a device.
SID	NUMBER	-	Identifier of the session where the prompt is created.
SERIAL#	NUMBER	-	Stores the unique serial number of the session in which the prompt was created.

DBMS_CLOUD_AI_AGENT Views

The DBMS_CLOUD_AI_AGENT package uses the following views.

Note

Support for the DBMS_CLOUD_AI_AGENT package is available in Oracle Database 19c starting with version 19.29, and in Oracle Database 26ai starting with version 23.26.

DBA_AI_AGENTS View

The view displays agents created by all users inside their database.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
AGENT_ID	NUMBER	NOT NULL	Stores the unique number assigned to an agent.

Column	Datatype	NULL	Description
AGENT_NAME	VARCHAR2(128)	NOT NULL	Specifies the agent name.
OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner of the agent.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the agent.
CREATED	TIMESTAMP(6) WITH TIME ZONE	-	The timestamp the agent was created.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	-	Indicates the timestamp of the most recent modification to the agent.
STATUS	VARCHAR2(8)	-	Specifies the status of the agent.
ORACLE_MAINTAINED	CHAR(1)	-	Indicates that the object is maintained by Oracle.

DBA_AI_AGENTS_ATTRIBUTES View

The view displays attributes of the agents created by all users inside their database.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
AGENT_ID	NUMBER	NOT NULL	Stores the unique number assigned to an agent.
AGENT_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent.
OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner of the agent attributes.
ATTRIBUTE_NAME	VARCHAR2(128)	-	Specifies the name of the attributes.
ATTRIBUTE_VALUE	CLOB	NOT NULL	Specifies the attribute value.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the agent attribute.

USER_AI_AGENTS View

The view displays details on conversations in your schema.

The current user can see the agents they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
AGENT_ID	NUMBER	NOT NULL	Stores the unique number assigned to an agent.
AGENT_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the agent.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the agent was created.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the agent.
STATUS	VARCHAR2(8)	-	Specifies the status of the agent.

Column	Datatype	NULL	Description
ORACLE_MAINTAINED	NUMBER	-	Indicates if it is maintained by Oracle.

USER_AI_AGENTS_ATTRIBUTES View

The view displays attributes of agents created by the current user inside their schema.

The current user can see the agent attributes they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
AGENT_ID	NUMBER	NOT NULL	Stores the unique number assigned to an agent.
AGENT_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent.
ATTRIBUTE_NAME	VARCHAR2(128)	-	Specifies the name of the attributes.
ATTRIBUTE_VALUE	CLOB	NOT NULL	Specifies the attribute value.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the agent attribute.

DBA_AI_AGENT_TOOLS View

The view displays details of the tools created by all users inside their database.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
TOOL_ID	NUMBER	NOT NULL	Stores the unique number assigned to a tool.
TOOL_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the tool.
OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner of the tool.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the tool.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the tool was created.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the tool.
STATUS	VARCHAR2(8)	-	Specifies the status of the tool.
ORACLE_MAINTAINED	CHAR(1)	-	Indicates if the object is maintained by Oracle.

DBA_AI_AGENT_TOOL_ATTRIBUTES View

The view displays attributes of tools created by all users inside their database.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
TOOL_ID	NUMBER	NOT NULL	Stores the unique number assigned to a tool attribute.
AGENT_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the tool.
OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner of the tool attribute.
ATTRIBUTE_NAME	VARCHAR2(128)	-	Specifies the name of the attribute.
ATTRIBUTE_VALUE	CLOB	NOT NULL	Specifies the attribute value.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the tool attributes.

USER_AI_AGENT_TOOLS View

The view displays details tools created by the current user inside their schema.

The current user can see the tools they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
TOOL_ID	NUMBER	NOT NULL	Stores the unique number assigned to a tool.
TOOL_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the tool.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the tool.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the tool was created.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the tool.
STATUS	VARCHAR2(8)	-	Specifies the status of the tool.
ORACLE_MAINTAINED	CHAR(1)	-	Indicates if the object is maintained by Oracle.

USER_AI_AGENT_TOOL_ATTRIBUTES View

The view displays attributes of tools created by the current user inside their schema.

The current user can see the tool attributes they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
TOOL_ID	NUMBER	NOT NULL	Stores the unique number assigned to a tool.
AGENT_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the tool.
ATTRIBUTE_NAME	VARCHAR2(128)	-	Specifies the name of the attribute.
ATTRIBUTE_VALUE	CLOB	NOT NULL	Specifies the attribute value.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the tool attributes.

DBA_AI_AGENT_TASK View

The view displays tasks created by all users inside their database.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
TASK_ID	NUMBER	NOT NULL	Stores the unique number assigned to a task.
TASK_NAME	VARCHAR2(128)	NOT NULL	Specifies the task name.
OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner of the task.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the task.
CREATED	TIMESTAMP(6) WITH TIME ZONE	-	The timestamp the task was created.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	-	Indicates the timestamp of the most recent modification to the task.
STATUS	VARCHAR2(8)	-	Specifies the status of the task.
ORACLE_MAINTAINED	CHAR(1)	-	Indicates that the object is maintained by Oracle.

DBA_AI_AGENT_TASK_ATTRIBUTES View

The view displays attributes of tasks created by all users inside their database.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
TASK_ID	NUMBER	NOT NULL	Stores the unique number assigned to a task.
TASK_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the task.
OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner of the task attributes.
ATTRIBUTE_NAME	VARCHAR2(128)	-	Specifies the name of the attributes.
ATTRIBUTE_VALUE	CLOB	NOT NULL	Specifies the attribute value.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the task attribute.

USER_AI_AGENTS View

The view displays details on tasks created by the current user inside their schema.

The current user can see the tasks they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
TASK_ID	NUMBER	NOT NULL	Stores the unique number assigned to a task.
TASK_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the task.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the task.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the task was created.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the task.

Column	Datatype	NULL	Description
STATUS	VARCHAR2(8)	-	Specifies the status of the task.
ORACLE_MAINTAINED	NUMBER	-	Indicates if the task is maintained by Oracle.

USER_AI_AGENT_TASK_ATTRIBUTES View

The view displays attributes of tasks created by the current user inside their schema.

The current user can see the agent task attributes they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
TASK_ID	NUMBER	NOT NULL	Stores the unique number assigned to an agent task.
TASK_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the task.
ATTRIBUTE_NAME	VARCHAR2(128)	-	Specifies the name of the attributes.
ATTRIBUTE_VALUE	CLOB	NOT NULL	Specifies the attribute value.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the agent task attribute.

DBA_AI_AGENT_TEAMS View

The view displays details of the teams created by all users inside their database.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
TEAM_ID	NUMBER	NOT NULL	Stores the unique number assigned to a tool.
TEAM_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the team.
OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner of the team.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the team.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the team was created.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the team.
STATUS	VARCHAR2(8)	-	Specifies the status of the team.
ORACLE_MAINTAINED	CHAR(1)	-	Indicates if the object is maintained by Oracle.

DBA_AI_AGENT_TEAM_ATTRIBUTES View

The view displays attributes of teams created by all users inside their database.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
TEAM_ID	NUMBER	NOT NULL	Stores the unique number assigned to a task.
TEAM_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the team.
OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner of the team attributes.
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the attributes.
ATTRIBUTE_VALUE	CLOB	-	Specifies the attribute value.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	-	Indicates the timestamp of the most recent modification to the team attribute.

USER_AI_AGENT_TEAMS View

The view displays details on teams created by the current user inside their schema.

The current user can see the team they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
TEAM_ID	NUMBER	NOT NULL	Stores the unique number assigned to an team.
TEAM_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the team.
DESCRIPTION	CLOB	-	Provides a description summarizing the purpose or context of the team.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	The timestamp the team was created.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Indicates the timestamp of the most recent modification to the team.
STATUS	VARCHAR2(8)	-	Specifies the status of the team.
ORACLE_MAINTAINED	NUMBER	-	Indicates if the object is maintained by Oracle.

USER_AI_AGENT_TEAM_ATTRIBUTES View

The view displays details of attributes of teams created by the current user inside their schema.

The current user can see the team attributes they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
TEAM_ID	NUMBER	NOT NULL	Stores the unique number assigned to an agent team.
TEAM_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the team.
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the attribute.
ATTRIBUTE_VALUE	CLOB	-	Specifies the attribute value.

Column	Datatype	NULL	Description
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	-	Indicates the timestamp of the most recent modification to the agent team attribute.

DBMS_CLOUD_AI_AGENT History Views

The DBMS_CLOUD_AI history views are categorized into team history view, task history view, and tool history views.

Note

Support for the DBMS_CLOUD_AI_AGENT package is available in Oracle Database 19c starting with version 19.29, and in Oracle Database 26ai starting with version 23.26.

See Also

[#unique_1158](#)

Team History Views

These views show all agent team-runs across the system. Each row records one agent team run, including team name, job names, start or end timestamps, run state, and any errors. It serves as the top-level run log, linking to task- and tool-level history for deeper analysis.

Task History Views

These views show agent task parameters within a team. These views holds the contextual data needed to run an agent task. The task reads input parameters from it when it starts and writes results back when it finishes. It connects the scheduling system with the agent framework.

Tool History Views

These views show calls to tools across the system. Each row logs the tool name, triggering agent and task, team execution ID, step name, timestamps, and input and output data. Use it to audit tool usage, monitor integrations, and debug tool behavior.

DBA_AI_AGENT_TEAM_HISTORY View

The view displays all agent team-runs across the system.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
TEAM_EXEC_ID	VARCHAR2(128)	NOT NULL	Stores the unique identifier for the team run instance, typically a GUID.
TEAM_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent being run.
TEAM_OWNER	VARCHAR2(128)	NOT NULL	Specifies the owner schema of the team.

Column	Datatype	NULL	Description
START_DATE	TIMESTAMP	-	The timestamp the process started.
END_DATE	TIMESTAMP	NOT NULL	The timestamp the process ended. If the process is still running, the value is NULL.
STATE	VARCHAR2(30)	NOT NULL	Indicates the current run state. Possible values are: - RUNNING: The agent team is actively processing the request. - WAITING_FOR_HUMAN: The agent team paused and is waiting for user input or approval. - RESUMING: The agent team continues after a pause or human input. - SUCCEEDED: The agent team finished the run and produced a final response. - FAILED: The agent team stopped due to an error or unmet condition.
TEAM_INSTANCE_ID	NUMBER	-	Specifies the session instance ID where the agent team was run.

USER_AI_AGENT_TEAM_HISTORY View

The view displays all agent team runs for teams owned by the current user.

The current user can see the team history they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
TEAM_EXEC_ID	VARCHAR2(128)	NOT NULL	Stores the unique identifier for the team execution instance, typically a GUID.
TEAM_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent being run.
START_DATE	TIMESTAMP	-	The timestamp the process started.
END_DATE	TIMESTAMP	NOT NULL	The timestamp the process ended. If the process is still running, the value is NULL.
STATE	VARCHAR2(30)	NOT NULL	Indicates the current run state. Possible values are: - RUNNING: The agent team is actively processing the request. - WAITING_FOR_HUMAN: The agent team paused and is waiting for user input or approval. - RESUMING: The agent team continues after a pause or human input. - SUCCEEDED: The agent team finished the run and produced a final response. - FAILED: The agent team stopped due to an error or unmet condition.
TEAM_INSTANCE_ID	NUMBER	-	Specifies the session instance ID where the agent team was run.

DBA_AI_AGENT_TASK_HISTORY View

This view shows agent task parameters within a team.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
TEAM_EXEC_ID	VARCHAR2(128)	NOT NULL	Stores the unique identifier for the team run instance, typically a GUID.
TEAM_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent team being run.
TASK_OWNER	VARCHAR2(128)		Specifies the owner schema of the team.
AGENT_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent being run.
TASK_NAME	VARCHAR2(128)		Indicates the name of the task being run.
TASK_ORDER	NUMBER		Identifies the current task's position in the agent team sequence. Because a task can appear multiple times, task_order distinguishes this specific occurrence.
CONVERSATION_PARAMS	CLOB		Provides JSON parameters for the conversation API, including: conversation_id, title, and title.
INPUT	CLOB		Specifies the input data for the task.
RESULT	CLOB		Stores the output produced by the task. (For example, text, JSON, rows, or a status message).
STATE	VARCHAR2(30)	NOT NULL	Indicates the current run state. Possible values are: - RUNNING: The agent team is actively processing the request. - WAITING_FOR_HUMAN: The agent team paused and is waiting for user input or approval. - RESUMING: The agent team continues after a pause or human input. - SUCCEEDED: The agent team finished the run and produced a final response. - FAILED: The agent team stopped due to an error or unmet condition.
START_DATE	TIMESTAMP	-	The timestamp the process started.
END_DATE	TIMESTAMP	NOT NULL	The timestamp the process ended. If the process is still running, the value is NULL.

USER_AI_AGENT_TASK_HISTORY View

This view shows agent task parameters for the current user's teams.

The current user can see the task history they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
TEAM_EXEC_ID	VARCHAR2(128)	NOT NULL	Stores the unique identifier for the team run instance, typically a GUID.
TEAM_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent team being run.
AGENT_NAME	VARCHAR2(128)	NOT NULL	Specifies the name of the agent being run.
TASK_NAME	VARCHAR2(128)		Indicates the name of the task being run.
TASK_ORDER	NUMBER		Identifies the current task's position in the agent team sequence. Because a task can appear multiple times, task_order distinguishes this specific occurrence.
CONVERSATION_PARAMETERS	CLOB		Provides JSON parameters for the conversation API, including: conversation_id, title, and title.
INPUT	CLOB		Specifies the input data for the task.
RESULT	CLOB		Stores the output produced by the task. (For example, text, JSON, rows, or a status message).
STATE	VARCHAR2(30)	NOT NULL	Indicates the current run state. Possible values are: - RUNNING: The agent team is actively processing the request. - WAITING_FOR_HUMAN: The agent team paused and is waiting for user input or approval. - RESUMING: The agent team continues after a pause or human input. - SUCCEEDED: The agent team finished the run and produced a final response. - FAILED: The agent team stopped due to an error or unmet condition.
START_DATE	TIMESTAMP	-	The timestamp the process started.
END_DATE	TIMESTAMP	NOT NULL	The timestamp the process ended. If the process is still running, the value is NULL.

DBA_AI_AGENT_TOOL_HISTORY View

This view lists calls to tools across the system.

Only the Database Administrator is privileged to access this view. They can grant access to other users. The DBA* views show objects of all Database users.

Column	Datatype	NULL	Description
invocation_id	VARCHAR2(128)	NOT NULL	Stores the unique identifier for the call.
TEAM_EXEC_ID	VARCHAR2(128)	NOT NULL	Stores the unique identifier for the team run instance, typically a GUID.
TASK_ORDER	NUMBER	-	Identifies the current task's position in the agent team sequence. Because a task can appear multiple times, task_order distinguishes this specific occurrence.

Column	Datatype	NULL	Description
TOOL_OWNER	VARCHAR2(128)	-	Owner schema of the tool.
TOOL_NAME	VARCHAR2(128)	-	Indicates the tool name that is called.
AGENT_NAME	VARCHAR2(128)	-	Indicates the agent name that is used for calling the tool.
TASK_NAME	VARCHAR2(128)	-	Name of the task that ran the tool.
START_DATE	TIMESTAMP	-	The timestamp when the tool started running.
END_DATE	TIMESTAMP	NOT NULL	The timestamp the when the tool stopped. If the process is still running, the value is NULL.
INPUT	CLOB	-	Specifies the input data for the tool.
OUTPUT	CLOB	-	Stores the output produced by the tool. (For example, text, JSON, rows, or a status message).
TOOL_OUTPUT	CLOB	-	Diagnostic messages from the tool function, used for debugging or troubleshooting and tracing behavior.

USER_AI_AGENT_TOOL_HISTORY View

This view lists calls to tools the current user owns.

The current user can see the tool history they own. You cannot grant access to other users. The Database Administrator can access these views but sees only the objects they own.

Column	Datatype	NULL	Description
invocation_id	VARCHAR2(128)	NOT NULL	Stores the unique identifier for the call.
TEAM_EXEC_ID	VARCHAR2(128)	NOT NULL	Stores the unique identifier for the team run instance, typically a GUID.
TASK_ORDER	NUMBER	-	Identifies the current task's position in the agent team sequence. Because a task can appear multiple times, task_order distinguishes this specific occurrence.
TOOL_NAME	VARCHAR2(128)	-	Indicates the tool name that is called.
AGENT_NAME	VARCHAR2(128)	-	Indicates the agent name that is used for calling the tool.
TASK_NAME	VARCHAR2(128)	-	Name of the task that ran the tool.
START_DATE	TIMESTAMP	-	The timestamp when the tool started running.
END_DATE	TIMESTAMP	NOT NULL	The timestamp the when the tool stopped. If the process is still running, the value is NULL.
INPUT	CLOB	-	Specifies the input data for the tool.
OUTPUT	CLOB	-	Stores the output produced by the tool. (For example, text, JSON, rows, or a status message).
TOOL_OUTPUT	CLOB	-	Diagnostic messages from the tool function, used for debugging or troubleshooting and tracing behavior.

DBMS_CLOUD_FUNCTION Views

The DBMS_CLOUD_FUNCTION package uses the following views.

- [DBA_CLOUD_FUNCTION View](#)

The DBA_CLOUD_FUNCTION view lists all the functions created in your Autonomous AI Database.

- [DBA_CLOUD_FUNCTION_CATALOG View](#)

The DBA_CLOUD_FUNCTION_CATALOG view lists all the catalogs created in an Autonomous AI Database instance.

- [USER_CLOUD_FUNCTION View](#)

The USER_CLOUD_FUNCTION view lists all the functions created by the user.

- [USER_CLOUD_FUNCTION_CATALOG View](#)

The USER_CLOUD_FUNCTION_CATALOG view lists all the catalogs created by the user.

- [USER_CLOUD_FUNCTION_ERRORS View](#)

The USER_CLOUD_FUNCTION_ERRORS view lists the errors generated during the connection validation to the remote library location.

DBA_CLOUD_FUNCTION View

The DBA_CLOUD_FUNCTION view lists all the functions created in your Autonomous AI Database.

Column	Datatype	Description
OWNER	VARCHAR2(128)	Owner of the catalog
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
APPLICATION_ID	VARCHAR2(2000)	OCI Application ID
CLOUD_FUNCTION_NAME	VARCHAR2(128)	Remote cloud function name
FUNCTION_NAME	VARCHAR2(128)	Function Name
FUNCTION_ID	VARCHAR2(2000)	OCI Function ID or AWS Function ARN
API_TYPE	VARCHAR2(128)	API type (Function or procedure)

DBA_CLOUD_FUNCTION_CATALOG View

The DBA_CLOUD_FUNCTION_CATALOG view lists all the catalogs created in an Autonomous AI Database instance.

Column	Datatype	Description
OWNER	VARCHAR2(128)	Owner of the catalog
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
CLOUD_PROVIDER	VARCHAR2(128)	OCI or AWS

USER_CLOUD_FUNCTION View

The USER_CLOUD_FUNCTION view lists all the functions created by the user.

Column	Datatype	Description
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
APPLICATION_NAME	VARCHAR2(128)	OCI Application Name
APPLICATION_ID	VARCHAR2(2000)	OCI Application ID
CLOUD_FUNCTION_NAME	VARCHAR2(128)	Remote cloud function name
FUNCTION_NAME	VARCHAR2(128)	Function Name
FUNCTION_ID	VARCHAR2(2000)	OCI Function ID or AWS Function ARN
API_TYPE	VARCHAR2(128)	API type (Function or procedure)

USER_CLOUD_FUNCTION_CATALOG View

The USER_CLOUD_FUNCTION_CATALOG view lists all the catalogs created by the user.

Column	Datatype	Description
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
CLOUD_PROVIDER	VARCHAR2(128)	OCI or AWS

USER_CLOUD_FUNCTION_ERRORS View

The USER_CLOUD_FUNCTION_ERRORS view lists the errors generated during the validation.

Column	Datatype	Description
CATALOG_NAME	VARCHAR2(128)	Name of the catalog
CREATE_TIMESTAMP	TIMESTAMP	Timestamp when error was generated
log_error_msg	CLOB	The error message

Pipeline Views

Lists details for the DBMS_CLOUD_PIPELINE views.

DBA_CLOUD_PIPELINES View

DBA_CLOUD_PIPELINES displays the list of pipelines created in the database. The view contains one row for each pipeline.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline.
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the pipeline
PIPELINE_TYPE	VARCHAR2(7)	NOT NULL	Type of pipeline: LOAD or EXPORT
STATUS	VARCHAR2(7)	NOT NULL	Current status of pipeline: STARTED or STOPPED
DESCRIPTION	VARCHAR2(4000)		User specified description for the pipeline.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation time for the pipeline.

Column	Datatype	NULL	Description
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the pipeline.
LAST_EXECUTION	TIMESTAMP(6) WITH TIME ZONE		Last successful execution time of pipeline job.
OPERATION_ID	NUMBER		Is the Operation ID for use with pipeline logs. See Monitor and Troubleshoot Pipelines for information on the USER_LOAD_OPERATIONS view for more information on data load operations.
STATUS_TABLE	VARCHAR2(128)		Status Table for pipeline progress. This is applicable for Load Pipelines.
ORACLE_MAINTAINED	CHAR(1)	NOT NULL	Indicates whether pipeline is Oracle maintained (Y) or not (N).

USER_CLOUD_PIPELINES View

USER_CLOUD_PIPELINES displays the list of pipelines created in the database for the user. The view contains one row for each pipeline.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline.
PIPELINE_TYPE	VARCHAR2(7)	NOT NULL	Type of pipeline: LOAD or EXPORT
STATUS	VARCHAR2(7)	NOT NULL	Current status of pipeline: STARTED or STOPPED
DESCRIPTION	VARCHAR2(4000)		User specified description for the pipeline.
CREATED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Creation time for the pipeline.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the pipeline.
LAST_EXECUTION	TIMESTAMP(6) WITH TIME ZONE		Last successful execution time of pipeline job.
OPERATION_ID	NUMBER		Is the Operation ID for use with pipeline logs. See Monitor and Troubleshoot Pipelines for information on the USER_LOAD_OPERATIONS view for more information on data load operations.
STATUS_TABLE	VARCHAR2(128)		Status Table for pipeline progress. This is applicable for Load Pipelines.
ORACLE_MAINTAINED	CHAR(1)	NOT NULL	Indicates whether pipeline is Oracle maintained (Y) or not (N).

DBA_CLOUD_PIPELINE_HISTORY View

DBA_CLOUD_PIPELINE_HISTORY lists the pipeline scheduled jobs in the database. Use this view to monitor the health of the pipeline and detect failures in pipeline job runs.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the pipeline.
PIPELINE_TYPE	VARCHAR2(7)		Type of pipeline: LOAD or EXPORT
JOB_ID	NUMBER	NOT NULL	Unique ID for the pipeline job execution.
JOB_NAME	VARCHAR2(1044)		Unique job name for the pipeline execution.
STATUS	VARCHAR2(30)		Status of the pipeline job.
START_DATE	TIMESTAMP(9) WITH TIME ZONE		Start time of the pipeline job execution.
END_DATE	TIMESTAMP(9) WITH TIME ZON		End time of the pipeline job execution.
INSTANCE_ID	NUMBER		Instance ID where the pipeline job ran.
SESSION_ID	VARCHAR2(128)		Session ID for the job session.
ERROR_NUMBER	NUMBER		Error number in the job (if encountered).
ERROR_MESSAGE	VARCHAR2(4000)		Error message in the job (if encountered).
CPU_USED	INTERVAL DAY(3) TO SECOND(2)		Number of CPU used by the pipeline job run.
JOB_OUTPUT	CLOB		Debug output of the pipeline job run.

USER_CLOUD_PIPELINE_HISTORY View

USER_CLOUD_PIPELINE_HISTORY lists the pipeline scheduled jobs in the database for the user. Use this view to monitor the health of the pipeline and detect failures in pipeline job runs.

Column	Data Type	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline
PIPELINE_TYPE	VARCHAR2(7)		Type of pipeline: LOAD or EXPORT
JOB_ID	NUMBER	NOT NULL	Unique ID for the pipeline job execution.
JOB_NAME	VARCHAR2(1044)		Unique job name for the pipeline execution.
STATUS	VARCHAR2(30)		Status of the pipeline job.
START_DATE	TIMESTAMP(9) WITH TIME ZONE		Start time of the pipeline job execution.
END_DATE	TIMESTAMP(9) WITH TIME ZON		End time of the pipeline job execution.
INSTANCE_ID	NUMBER		Instance ID where the pipeline job ran.
SESSION_ID	VARCHAR2(128)		Session ID for the job session.
ERROR_NUMBER	NUMBER		Error number in the job (if encountered).
ERROR_MESSAGE	VARCHAR2(4000)		Error message in the job (if encountered).
CPU_USED	INTERVAL DAY(3) TO SECOND(2)		Number of CPU used by the pipeline job run.
JOB_OUTPUT	CLOB		Debug output of the pipeline job run.

DBA_CLOUD_PIPELINE_ATTRIBUTES View

DBA_CLOUD_PIPELINE_ATTRIBUTES lists the attributes of a pipelines in the database. Attributes control the configuration and behavior of the pipeline jobs.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline.
OWNER	VARCHAR2(128)	NOT NULL	Owner schema of the pipeline
PIPELINE_TYPE	VARCHAR2(7)	NOT NULL	Type of pipeline: LOAD or EXPORT
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline attribute.
ATTRIBUTE_VALUE	CLOB		Value of the pipeline attribute.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the pipeline attribute.

USER_CLOUD_PIPELINE_ATTRIBUTES View

USER_CLOUD_PIPELINE_ATTRIBUTES lists the attributes of a pipelines in the database for the user. Attributes control the configuration and behavior of the pipeline.

Column	Datatype	NULL	Description
PIPELINE_ID	NUMBER	NOT NULL	Unique number assigned to the pipeline.
PIPELINE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline.
PIPELINE_TYPE	VARCHAR2(7)	NOT NULL	Type of pipeline: LOAD or EXPORT
ATTRIBUTE_NAME	VARCHAR2(128)	NOT NULL	Name of the pipeline attribute.
ATTRIBUTE_VALUE	CLOB		Value of the pipeline attribute.
LAST_MODIFIED	TIMESTAMP(6) WITH TIME ZONE	NOT NULL	Last modification time for the pipeline attribute.

Track DBMS_CLOUD Load Operations

DBMS_CLOUD tracks details for load operations for each instance and for users.

DBA_LOAD_OPERATIONS and USER_LOAD_OPERATIONS Views

USER_LOAD_OPERATIONS and DBA_LOAD_OPERATIONS views show the load operations in your schema or for all users. The USER_LOAD_OPERATIONS view does not display the USERNAME column.

Column	Datatype	NULL	Description
ID	NUMBER	NOT NULL	Specifies the operation ID.
TYPE	VARCHAR2(128)	NOT NULL	Shows the operation type.
SID	NUMBER	NOT NULL	Session ID of the session.
SERIAL#	NUMBER	NOT NULL	

Column	Datatype	NULL	Description
USERNAME	VARCHAR2(128)	NOT NULL	User ID of the user performing the operation. The USER_LOAD_OPERATIONS view does not display the USERNAME column.
START_TIME	TIMESTAMP(6) WITH TIME ZONE		Start time of the task.
UPDATE_TIME	TIMESTAMP(6) WITH TIME ZONE		Last update time for the task.
STATUS	VARCHAR2(9)		Status of the operation. Valid values are: COMPLETED, FAILED, PENDING, RUNNING, SKIPPED.
OWNER_NAME	VARCHAR2(128)		Schema that owns the operation.
TABLE_NAME	VARCHAR2(128)		
PARTITION_NAME	VARCHAR2(128)		
SUBPARTITION_NAME	VARCHAR2(128)		
FILE_URI_LIST	VARCHAR2(4000)		This value shows either a comma-delimited list of source file URIs or one or more directories and source files that were specified in the FILE_URI_LIST parameter.
ROWS_LOADED	NUMBER		Number of rows loaded in the load operation.
LOGFILE_TABLE	VARCHAR2(128)		Shows the name of the table you can query to look at the log of a load operation.
BADFILE_TABLE	VARCHAR2(128)		Shows the name of the table you can query to look at the rows that got errors during loading.
STATUS_TABLE	VARCHAR2(128)		Shows the name of the status table for the load operation.
TEMPEXT_TABLE	VARCHAR2(128)		
CREDENTIAL_NAME	VARCHAR2(128)		
EXPIRATION_TIME	TIMESTAMP(9) WITH TIME ZONE		

Events for Autonomous AI Database on Dedicated Exadata Infrastructure

The Oracle Database resources emit events, which are structured messages that indicate changes in resources. This article lists the operations that generate events for Autonomous AI Database on Oracle Cloud.

Autonomous AI Database generates events for more than forty different operations, providing you fine-grained choices for resource tracking. Following are the different event types that Autonomous AI Database infrastructure resources emit:

- Exadata Infrastructure Event Types
- Cloud Exadata Infrastructure Event Types
- Autonomous Exadata VM Cluster Event Types
- Cloud Autonomous VM Cluster Event Types

- Autonomous Container Database Event Types
- Autonomous AI Database Event Types
- Autonomous Data Guard Event Types

① Note

All the normal service events, metrics and logs that apply to Oracle Public Cloud also apply to Multicloud deployments. Additionally, for AWS Multicloud deployments you can use the monitoring and management services in Oracle Database@AWS. See [Monitoring services in Oracle Database@AWS](#) for details.

Exadata Infrastructure Event Types

APPLIES TO:  Exadata Cloud@Customer only

Please refer to the following pages for a comprehensive list of event types emitted by Exadata Infrastructure instances:

- Exadata Infrastructure Event Types
- Exadata Infrastructure Maintenance Event Types

Cloud Exadata Infrastructure Event Types

APPLIES TO:  Oracle Public Cloud only

Please refer to the following pages for a comprehensive list of event types emitted by Cloud Exadata Infrastructure instances:

- Cloud Exadata Infrastructure Event Types
- Cloud Exadata Infrastructure Maintenance Event Types
- Cloud Exadata Infrastructure Critical and Information Event Types

Autonomous Exadata VM Cluster Event Types

APPLIES TO:  Exadata Cloud@Customer only

Review the list of event types that Autonomous Exadata VM Clusters emit.

Friendly Name	Event Type
Autonomous VM Cluster - Change Compartment	com.oraclecloud.databaseservice.changeautonomousvmclustercompartment
Autonomous VM Cluster - Create Begin	com.oraclecloud.databaseservice.createautonomousvmcluster.begin
Autonomous VM Cluster - Create End	com.oraclecloud.databaseservice.createautonomousvmcluster.end
Autonomous VM Cluster - Critical See Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomousvmcluster.critical
Autonomous VM Cluster - Information See Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomousvmcluster.information

Friendly Name	Event Type
Autonomous VM Cluster - maintenance begin	com.oraclecloud.databaseservice.autonomous.autonomousscloudvmcluster.maintenance.begin
Autonomous VM Cluster - maintenance end	com.oraclecloud.databaseservice.autonomous.autonomousscloudvmcluster.maintenance.end
Autonomous VM Cluster - maintenance reminder	com.oraclecloud.databaseservice.autonomous.autonomousscloudvmcluster.maintenance.reminder
Autonomous VM Cluster - maintenance scheduled	com.oraclecloud.databaseservice.autonomous.autonomousscloudvmcluster.maintenance.scheduled
Autonomous VM Cluster - Terminate Begin	com.oraclecloud.databaseservice.deleteautonomoussvmcluster.begin
Autonomous VM Cluster - Terminate End	com.oraclecloud.databaseservice.deleteautonomoussvmcluster.end
Autonomous VM Cluster - Update Begin	com.oraclecloud.databaseservice.updateautonomoussvmcluster.begin
Autonomous VM Cluster - Update End	com.oraclecloud.databaseservice.updateautonomoussvmcluster.end
Autonomous VM Cluster - Warning See Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomous.cloudautonomoussvmcluster.warning

Example 7-1 Autonomous VM Cluster - Change Compartment

```
{
  "cloudEventsVersion": "0.1",
  "eventID": "60600c06-d6a7-4e85-b56a-1de3e6042f57",
  "eventType":
"com.oraclecloud.databaseservice.changeautonomoussvmclustercompartment",
  "source": "databaseservice",
  "eventTypeVersion": "1.0",
  "eventTime": "2019-06-27T21:16:04Z",
  "contentType": "application/json",
  "extensions": {
    "compartmentId": "ocid1.compartment.oc1..unique_ID"
  },
  "data": {
    "compartmentId": "ocid1.compartment.oc1..unique_ID",
    "compartmentName": "example_name",
    "resourceName": "my_database",
    "resourceId": "AutonomousVmCluster-unique_ID",
    "availabilityDomain": "all",
    "freeFormTags": {},
    "definedTags": {},
    "additionalDetails": {
      "id": "ocid1.id..oc1...unique_ID",
      "lifecycleState": "AVAILABLE",
      "timeCreated": "2019-09-03T12:00:00.000Z",
      "timeUpdated": "2019-09-03T12:30:00.000Z",
      "displayName": "testDisplayName",

```

```

        "lifecycleDetails": "detail message",
        "exadataInfrastructureId": "ExatraInfra-unique_ID",
        "vmClusterNetworkId": "VmClusterNetwork-unique_ID",
        "cpuCoreCount": "2",
        "availableCpus": "1",
        "dataStorageSizeInTBs": "4",
        "availableDataStorageSizeInTBs": "1",
        "licenseType": "BRING_YOUR_OWN_LICENSE",
        "timeZone": "US/Pacific"
    }
}
}

```

Example 7-2 Autonomous VM Cluster - Create Begin

```

{
  "cloudEventsVersion": "0.1",
  "eventId": "60600c06-d6a7-4e85-b56a-1de3e6042f57",
  "eventType":
"com.oraclecloud.databaseservice.createautonomousvmcluster.begin",
  "source": "databaseservice",
  "eventTypeVersion": "1.0",
  "eventTime": "2019-06-27T21:16:04Z",
  "contentType": "application/json",
  "extensions": {
    "compartmentId": "ocid1.compartment.oc1..unique_ID"
  },
  "data": {
    "compartmentId": "ocid1.compartment.oc1..unique_ID",
    "compartmentName": "example_name",
    "resourceName": "my_database",
    "resourceId": "AutonomousVmCluster-unique_ID",
    "availabilityDomain": "all",
    "freeFormTags": {},
    "definedTags": {},
    "additionalDetails": {
      "id": "ocid1.id..oc1..unique_ID",
      "lifecycleState": "AVAILABLE",
      "timeCreated": "2019-09-03T12:00:00.000Z",
      "timeUpdated": "2019-09-03T12:30:00.000Z",
      "displayName": "testDisplayName",
      "lifecycleDetails": "detail message",
      "exadataInfrastructureId": "ExatraInfra-unique_ID",
      "vmClusterNetworkId": "VmClusterNetwork-unique_ID",
      "cpuCoreCount": "2",
      "availableCpus": "1",
      "dataStorageSizeInTBs": "4",
      "availableDataStorageSizeInTBs": "1",
      "licenseType": "BRING_YOUR_OWN_LICENSE",
      "timeZone": "US/Pacific"
    }
  }
}
}

```


Example 7-3 Autonomous VM Cluster - Terminate Begin

```
{
  "cloudEventsVersion": "0.1",
  "eventID": "60600c06-d6a7-4e85-b56a-1de3e6042f57",
  "eventType":
"com.oraclecloud.databaseservice.deleteautonomousvmcluster.begin",
  "source": "databaseservice",
  "eventTypeVersion": "1.0",
  "eventTime": "2019-06-27T21:16:04Z",
  "contentType": "application/json",
  "extensions": {
    "compartmentId": "ocid1.compartment.oc1..unique_ID"
  },
  "data": {
    "compartmentId": "ocid1.compartment.oc1..unique_ID",
    "compartmentName": "example_name",
    "resourceName": "my_database",
    "resourceId": "AutonomousVmCluster-unique_ID",
    "availabilityDomain": "all",
    "freeFormTags": {},
    "definedTags": {},
    "additionalDetails": {
      "id": "ocid1.id..oc1...unique_ID",
      "lifecycleState": "AVAILABLE",
      "timeCreated": "2019-09-03T12:00:00.000Z",
      "timeUpdated": "2019-09-03T12:30:00.000Z",
      "displayName": "testDisplayName",
      "lifecycleDetails": "detail message",
      "exadataInfrastructureId": "ExatraInfra-unique_ID",
      "vmClusterNetworkId": "VmClusterNetwork-unique_ID",
      "cpuCoreCount": "2",
      "availableCpus": "1",
      "dataStorageSizeInTBs": "4",
      "availableDataStorageSizeInTBs": "1",
      "licenseType": "BRING_YOUR_OWN_LICENSE",
      "timeZone": "US/Pacific"
    }
  }
}
```

Example 7-4 Autonomous VM Cluster - Update Begin

```
{
  "cloudEventsVersion": "0.1",
  "eventID": "60600c06-d6a7-4e85-b56a-1de3e6042f57",
  "eventType":
"com.oraclecloud.databaseservice.updateautonomousvmcluster.begin",
  "source": "databaseservice",
  "eventTypeVersion": "1.0",
  "eventTime": "2019-06-27T21:16:04Z",
  "contentType": "application/json",
  "extensions": {
    "compartmentId": "ocid1.compartment.oc1..unique_ID"
  },
}
```

```

"data": {
  "compartmentId": "ocid1.compartment.oc1..unique_ID",
  "compartmentName": "example_name",
  "resourceName": "my_database",
  "resourceId": "AutonomousVmCluster-unique_ID",
  "availabilityDomain": "all",
  "freeFormTags": {},
  "definedTags": {},
  "additionalDetails": {
    "id": "ocid1.id..oc1...unique_ID",
    "lifecycleState": "AVAILABLE",
    "timeCreated": "2019-09-03T12:00:00.000Z",
    "timeUpdated": "2019-09-03T12:30:00.000Z",
    "displayName": "testDisplayName",
    "lifecycleDetails": "detail message",
    "exadataInfrastructureId": "ExatraInfra-unique_ID",
    "vmClusterNetworkId": "VmClusterNetwork-unique_ID",
    "cpuCoreCount": "2",
    "availableCpus": "1",
    "dataStorageSizeInTBs": "4",
    "availableDataStorageSizeInTBs": "1",
    "licenseType": "BRING_YOUR_OWN_LICENSE",
    "timeZone": "US/Pacific"
  }
}
}

```

Cloud Autonomous Exadata VM Cluster Event Types

APPLIES TO:  Oracle Public Cloud only

Review the list of event types that cloud Autonomous Exadata VM Clusters emit.

Friendly Name	Event Type
Autonomous Cloud VM Cluster - Change Compartment Begin	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.changecompartment.begin
Autonomous Cloud VM Cluster - Change Compartment End	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.changecompartment.end
Autonomous Cloud VM Cluster - Create Begin	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.create.begin
Autonomous Cloud VM Cluster - Create End	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.create.end
Autonomous Cloud VM Cluster - Critical See Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomous.cloudautonomouvmcluster.critical
Autonomous Cloud VM Cluster - Information See Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomous.cloudautonomouvmcluster.information
Autonomous Cloud VM Cluster - maintenance begin	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.maintenance.begin

Friendly Name	Event Type
Autonomous Cloud VM Cluster - maintenance end	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.maintenance.end
Autonomous Cloud VM Cluster - maintenance reminder	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.maintenance.reminder
Autonomous Cloud VM Cluster - maintenance scheduled	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.maintenance.scheduled
Autonomous Cloud VM Cluster - Terminate Begin	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.terminate.begin
Autonomous Cloud VM Cluster - Terminate End	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.terminate.end
Autonomous Cloud VM Cluster - Update Begin	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.update.begin
Autonomous Cloud VM Cluster - Update End	com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.update.end
Autonomous Cloud VM Cluster - Warning See Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomous.cloudautonomouvmcluster.warning

Example 7-5 Cloud Autonomous Exadata VM Cluster Event

```
{
  "cloudEventsVersion": "0.1",
  "eventID": "60600c06-d6a7-4e85-b56a-1de3e6042f57",
  "eventType":
"com.oraclecloud.databaseservice.autonomous.autonomoucloudvmcluster.update.be
gin",
  "source": "databaseservice",
  "eventTypeVersion": "1.0",
  "eventTime": "2020-06-02T21:16:04.000Z",
  "contentType": "application/json",
  "extensions": {
    "compartmentId": "ocid1.compartment.oc1..unique_id"
  },
  "data": {
    "compartmentId": "ocid1.compartment.oc1..unique_id",
    "compartmentName": "example_name",
    "resourceName": "my_container_database",
    "resourceId": "ocid1.autonomousexainfrastructure-..unique_id",
    "availabilityDomain": "all",
    "freeFormTags": {},
    "definedTags": {},
    "additionalDetails": {
      "lifecycleState": "ACTIVE",
      "timeCreated": "2019-06-27T21:15:59.000Z",
      "timeUpdated": "2019-06-27T21:16:04.389Z",
      "dbName": "exampleDbName"
    }
  }
}
```

```

    }
  }
}

```

Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types

Autonomous Exadata VM Clusters emit critical, warning, and information event types. These event types allow you to receive notifications:

- When your Autonomous Exadata VM Cluster needs urgent attention (critical events)
- For an event that is not critical but needs your attention to prevent it from turning critical (warning events)
- For an event that is good to be notified for your information (information events)

The eventType values for each of these events are listed below:

Autonomous Exadata VM Cluster Critical, Warning, and Informational Event Types

APPLIES TO:  Exadata Cloud@Customer only

Friendly Name	Event Type
Autonomous VM Cluster - Critical	com.oraclecloud.databaseservice.autonomous.vmcluster.critical
Autonomous VM Cluster - Warning	com.oraclecloud.databaseservice.autonomous.vmcluster.warning
Autonomous VM Cluster - Information	com.oraclecloud.databaseservice.autonomous.vmcluster.information

Cloud Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types

APPLIES TO:  Oracle Public Cloud only

Friendly Name	Event Type
Autonomous Cloud VM Cluster - Critical	com.oraclecloud.databaseservice.autonomous.cloudautonomousvmcluster.critical
Autonomous Cloud VM Cluster - Warning	com.oraclecloud.databaseservice.autonomous.cloudautonomousvmcluster.warning
Autonomous Cloud VM Cluster - Information	com.oraclecloud.databaseservice.autonomous.cloudautonomousvmcluster.information

These events use the additionalDetails section of the event message to provide specific details about what is happening within the Autonomous Exadata VM Cluster resource emitting the event. In the additionalDetails section, the eventName field provides the name of the critical, warning, or information event.

In the tables below, you can read about the conditions and operations that trigger critical, warning, and information events. Each condition or operation is identified by a unique eventName value.

Critical Events for Autonomous Exadata VM Cluster and Cloud Autonomous Exadata VM Cluster

Event	Description
sslcertificate.expired	This event is generated when the SSL Certificate expires. All the Autonomous AI Database wallets related to this Autonomous Exadata VM Cluster will expire.
autonomous.operations.issue.detected.critical	Autonomous Dedicated Operations has detected a critical issue with your Autonomous Exadata VM Cluster and is working on it. No action is required on your part.
autonomous.operations.issue.resolved.critical	Autonomous Dedicated Operations has resolved a critical issue with your Autonomous Exadata VM Cluster. No further action is required.
sslcertificate.rotated	This event is generated when the SSL certificate is rotated either manually (using the Oracle Cloud Infrastructure console or API) or automatically on its expiry.
storage.status.free_space	One of the diskgroups for your Autonomous Exadata VM Cluster is almost out of space. It's recommended that you increase the storage allocated to the Autonomous Exadata VM Cluster that received the alert.
dbnode.status.critical	Node : <nodename> is offline. The Autonomous AI Database operations team is currently investigating the issue and will provide more information as needed.

Warning Events for Autonomous Exadata VM Cluster and Cloud Autonomous Exadata VM Cluster

Event	Description
sslcertificateexpiry.reminder	This event is generated when an Autonomous Exadata VM Cluster determines that a wallet is due to expire in less than six (6) weeks. This event is reported at most once per week. This event is triggered when there is a connection that uses the wallet that is due to expire.
sslcertificaterotation.reminder	This event is generated when an SSL certificate is older than 365 days and recommends the customer to rotate the certificate. After an SSL certificate crosses 365 days, this reminder is once per week until it is rotated.
autonomous.operations.issue.detected.warning	Autonomous Dedicated Operations has detected an issue with your Autonomous Exadata VM Cluster and is working on it. No action is required on your part.
autonomous.operations.issue.resolved.warning	Autonomous Dedicated Operations has resolved an issue with your Autonomous Exadata VM Cluster. No further action is required.

Information Events for Autonomous Exadata VM Cluster and Cloud Autonomous Exadata VM Cluster

Event	Description
autonomous.operations.issue.detected.information	Autonomous Dedicated Operations has detected an issue with your Autonomous Exadata VM Cluster and is working on it. No action is required on your part.
autonomous.operations.issue.resolved.information	Autonomous Dedicated Operations has resolved an issue with your Autonomous Exadata VM Cluster. No further action is required.
storage.status.free_space_cleared	The lack of free space in one of the diskgroups for your Autonomous Exadata VM Cluster has been resolved.
dbnode.status.information	Node : <nodename> is now online.

Autonomous Container Database Event Types

Review the list of event types that Autonomous Container Databases emit.

Friendly Name	Event Type
Autonomous Container Database - Add Standby Begin	com.oraclecloud.databaseservice.autonomous.container.database.dataguard.createstandby.begin
Autonomous Container Database - Add Standby End	com.oraclecloud.databaseservice.autonomous.container.database.dataguard.createstandby.end
Autonomous Container Database - Change Compartment	com.oraclecloud.databaseservice.changeautonomouscontainerdatabasecompartment
Autonomous Container Database - Convert To Physical Standby Begin	com.oraclecloud.databaseservice.autonomous.container.database.physical.standby.conversion.begin
Autonomous Container Database - Convert To Physical Standby End	com.oraclecloud.databaseservice.autonomous.container.database.physical.standby.conversion.end
Autonomous Container Database - Convert To Physical Standby Reminder	com.oraclecloud.dbaas.autonomous.container.database.automatic.conversion.to.physical.standby.reminder
Autonomous Container Database - Convert To Snapshot Standby Begin	com.oraclecloud.databaseservice.autonomous.container.database.snapshot.standby.conversion.begin
Autonomous Container Database - Convert To Snapshot Standby End	com.oraclecloud.databaseservice.autonomous.container.database.snapshot.standby.conversion.end
Autonomous Container Database - Create Backup Begin	com.oraclecloud.databaseservice.autonomous.container.database.backup.begin
Autonomous Container Database - Create Backup End	com.oraclecloud.databaseservice.autonomous.container.database.backup.end
Autonomous Container Database - Create Begin	com.oraclecloud.databaseservice.autonomous.container.database.instance.create.begin

Friendly Name	Event Type
Autonomous Container Database - Create End	com.oraclecloud.databaseservice.autonomous.container.database.instance.create.end
Autonomous Container Database - Critical See Autonomous Container Database Critical, Warning, and Information Event Types	com.oraclecloud.databaseservice.autonomous.container.database.critical
Autonomous Container Database - Failover Begin	com.oraclecloud.databaseservice.autonomous.container.database.dataguard.failover.begin
Autonomous Container Database - Failover End	com.oraclecloud.databaseservice.autonomous.container.database.dataguard.failover.end
Autonomous Container Database - Information See Autonomous Container Database Critical, Warning, and Information Event Types	com.oraclecloud.databaseservice.autonomous.container.database.information
Autonomous Container Database - Maintenance Begin	com.oraclecloud.databaseservice.autonomous.container.database.maintenance.begin
Autonomous Container Database - Maintenance End	com.oraclecloud.databaseservice.autonomous.container.database.maintenance.end
Autonomous Container Database - Maintenance Reminder	com.oraclecloud.databaseservice.autonomous.container.database.maintenance.reminder
Autonomous Container Database - Maintenance Scheduled	com.oraclecloud.databaseservice.autonomous.container.database.maintenance.scheduled
Autonomous Container Database - Reinstatement Begin	com.oraclecloud.databaseservice.autonomous.container.database.dataguard.reinstatement.begin
Autonomous Container Database - Reinstatement End	com.oraclecloud.databaseservice.autonomous.container.database.dataguard.reinstatement.end
Autonomous Container Database - Restart Begin	com.oraclecloud.databaseservice.restart.autonomouscontainerdatabase.begin
Autonomous Container Database - Restart End	com.oraclecloud.databaseservice.restart.autonomouscontainerdatabase.end
Autonomous Container Database - Restore Begin	com.oraclecloud.databaseservice.autonomous.container.database.restore.begin
Autonomous Container Database - Restore End	com.oraclecloud.databaseservice.autonomous.container.database.restore.end
Autonomous Container Database - Rotate Encryption Key Begin	com.oraclecloud.databaseservice.rotateautonomouscontainerdatabaseencryptionkey.begin
Autonomous Container Database - Rotate Encryption Key End	com.oraclecloud.databaseservice.rotateautonomouscontainerdatabaseencryptionkey.end
Autonomous Container Database - Switchover Begin	com.oraclecloud.databaseservice.autonomous.container.database.dataguard.switchover.begin

Friendly Name	Event Type
Autonomous Container Database - Switchover End	com.oraclecloud.databaseservice.autonomous.container.database.dataguard.switchover.end
Autonomous Container Database - Terminate Begin	com.oraclecloud.databaseservice.terminateautonomouscontainerdatabase.begin
Autonomous Container Database - Terminate End	com.oraclecloud.databaseservice.terminateautonomouscontainerdatabase.end
Autonomous Container Database - Update Begin	com.oraclecloud.databaseservice.autonomous.container.database.instance.update.begin
Autonomous Container Database - Update End	com.oraclecloud.databaseservice.autonomous.container.database.instance.update.end
Autonomous Container Database - Warning See Autonomous Container Database Critical, Warning, and Information Event Types	com.oraclecloud.databaseservice.autonomous.container.database.warning

Example 7-6 Autonomous Container Database Event

```
{
  "cloudEventsVersion": "0.1",
  "eventID": "<unique_ID>",
  "eventType":
"com.oraclecloud.databaseservice.autonomous.container.database.backup.begin",
  "source": "databaseservice",
  "eventTypeVersion": "2.0",
  "eventTime": "2019-06-27T21:16:04Z",
  "contentType": "application/json",
  "extensions": {
    "compartmentId": "ocid1.compartment.oc1..<unique_ID>"
  },
  "data": {
    "compartmentId": "ocid1.compartment.oc1..<unique_ID>",
    "compartmentName": "example_name",
    "resourceName": "my_container_database",
    "resourceId": "<unique_ID>",
    "availabilityDomain": "all",
    "freeFormTags": {},
    "definedTags": {},
    "additionalDetails": {
      "cpuCoreCount": null,
      "lifecycleState": "ACTIVE",
      "dataStorageSizeInTBs": null,
      "timeCreated": "2019-06-27T21:15:59.000Z",
      "timeUpdated": "2019-06-27T21:16:04.389Z",
      "dbUniqueName": "dwrrdtsr_phx289",
      "dbHomeId": "ocid1.autonomoushome.oc1.phx.<unique_ID>",
      "dbName": "dwrrdtsr"
    },
    "autonomousContainerDatabaseId":
"ocid1.autonomouscontainerdatabase.oc1.phx.<unique_ID>"
  }
}
```



```
}
}
```

Autonomous Container Database Critical, Warning, and Information Event Types

Autonomous Container Databases emit critical, warning, and information event types. These event types allow you to receive notifications:

- When your Autonomous Container Database needs urgent attention (critical events)
- For an event that is not critical but needs your attention to prevent it from turning critical (warning events)
- For an event that is good to be notified for your information (information events)

The eventType values for each of these events are listed below:

Friendly Name	Event Type
Autonomous Container Database - Critical	com.oraclecloud.databaseservice.autonomous.container.database.critical
Autonomous Container Database - Warning	com.oraclecloud.databaseservice.autonomous.container.database.warning
Autonomous Container Database - Information	com.oraclecloud.databaseservice.autonomous.container.database.information

These events use the additionalDetails section of the event message to provide specific details about what is happening within the Autonomous Container Database resource emitting the event. In the additionalDetails section, the eventName field provides the name of the critical, warning, or information event.

In the tables below, you can read about the conditions and operations that trigger critical, warning, and information events. Each condition or operation is identified by a unique eventName value.

Critical Events for Autonomous Container Database

Event	Description
autonomous.operations.issue.detected.critical	Autonomous Dedicated Operations has detected a critical issue with your Autonomous Container Database and is working on it. No action is required on your part.
autonomous.operations.issue.resolved.critical	Autonomous Dedicated Operations has resolved a critical issue with your Autonomous Container Database. No further action is required.
backup.destination.error	APPLIES TO:  Exadata Cloud@Customer only Autonomous Dedicated Operations has detected one of the following critical issues with your backup destination: • Your Network File Storage server is out of space. • Incorrect permissions are set on your Network File Storage server. • Your Zero Data Loss Recovery Appliance server is out of space.

Event	Description
database.status.archiver_hang	Operations has been notified and will investigate the issue. More information will be sent via Autonomous Container Database events if user assistance is required.
databasevault.disabled	Database Vault for this Autonomous AI Database was disabled by a process other than standard Oracle patching automation. Please review your database security configuration and audit log
databasevault.enabled	Database Vault for this Autonomous AI Database was enabled by a process other than standard Oracle patching automation. Please review your database security configuration and audit logs.
fra.status.free_space	Database: <cdbName> FRA usage is in the range of 90% to 100%.

Warning Events for Autonomous Container Database

Event	Description
autonomous.operations.issue.detected.warning	Autonomous Dedicated Operations has detected an issue with your Autonomous Container Database and is working on it. No action is required on your part.
autonomous.operations.issue.resolved.warning	Autonomous Dedicated Operations has resolved an issue with your Autonomous Container Database. No further action is required.
backup.destination.nfs.limit.breach	APPLIES TO:  Exadata Cloud@Customer only Generated when Network File Storage backups consume at least 80% of space in the backup destination. If space usage continues to rise after 80 percent, Ops may send manual critical events.

Information Events for Autonomous Container Database

Event	Description
autonomous.operations.issue.detected.information	Autonomous Dedicated Operations has detected an issue with your Autonomous Container Database and is working on it. No action is required on your part.
autonomous.operations.issue.resolved.information	Autonomous Dedicated Operations has resolved an issue with your Autonomous Container Database. No further action is required.
backup.validation.complete	Your Autonomous Container Database's backup validation is complete.
backup.destination.available	APPLIES TO:  Exadata Cloud@Customer only Your Autonomous Container Database's backup destination is back to a healthy state.
fra.status.free_space_cleared	Database: <cdbName> FRA usage is no longer in the range of 90% to 100%.

Autonomous AI Database Event Types

Review the list of event types that Autonomous AI Databases emit.

Autonomous AI Database Event Types

Friendly Name	Event Type
Autonomous AI Database - Automatic Backup Begin	com.oraclecloud.databaseservice.automaticbackupautonomousdatabase.begin
Autonomous AI Database - Automatic Backup End	com.oraclecloud.databaseservice.automaticbackupautonomousdatabase.end
Autonomous AI Database - Backup Update – Begin	com.oraclecloud.databaseservice.autonomous.database.backup.updateautonomousdatabasebackup.begin
Autonomous AI Database - Backup Update – End	com.oraclecloud.databaseservice.autonomous.database.backup.updateautonomousdatabasebackup.end
Autonomous AI Database - Change Compartment Begin	com.oraclecloud.databaseservice.changeautonomousdatabasecompartment.begin
Autonomous AI Database - Change Compartment End	com.oraclecloud.databaseservice.changeautonomousdatabasecompartment.end
Autonomous AI Database - Change Database Name Begin	com.oraclecloud.databaseservice.changeautonomousdatabasename.begin
Autonomous AI Database - Change Database Name End	com.oraclecloud.databaseservice.changeautonomousdatabasename.end
Autonomous AI Database - Create Backup Begin	com.oraclecloud.databaseservice.autonomous.database.backup.begin
Autonomous AI Database - Create Backup End	com.oraclecloud.databaseservice.autonomous.database.backup.end
Autonomous AI Database - Create Begin	com.oraclecloud.databaseservice.autonomous.database.instance.create.begin
Autonomous AI Database - Create End	com.oraclecloud.databaseservice.autonomous.database.instance.create.end
Autonomous AI Database - Critical See Autonomous AI Database Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomous.database.critical
Autonomous AI Database - Information See Autonomous AI Database Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomous.database.information
Autonomous AI Database - Deregister Autonomous AI Database with Data Safe Begin	com.oraclecloud.databaseservice.deregisterautonomousdatabasedatasafe.begin
Autonomous AI Database - Deregister Autonomous AI Database with Data Safe End	com.oraclecloud.databaseservice.deregisterautonomousdatabasedatasafe.end
Autonomous AI Database - Generate Wallet	com.oraclecloud.databaseservice.generateautonomousdatabasewallet
Autonomous AI Database - Long-term Backup Will Expire In 24 Hours	com.oraclecloud.databaseservice.autonomous.database.backup.longtermbackupexpiresinaday.reminder

Friendly Name	Event Type
Autonomous AI Database - Long-term Backup Will Expire In 30 Days	com.oraclecloud.databaseservice.autonomous.database.backup.longtermbackupexpiresinamonth.reminder
Autonomous AI Database - Long-term Backup Will Expire In 90 Days	com.oraclecloud.databaseservice.autonomous.database.backup.longtermbackupexpiresinthree months.reminder
Autonomous AI Database - Register Autonomous AI Database with Data Safe Begin	com.oraclecloud.databaseservice.registerautonomousdatabasedatasafe.begin
Autonomous AI Database - Register Autonomous AI Database with Data Safe End	com.oraclecloud.databaseservice.registerautonomousdatabasedatasafe.end
Autonomous AI Database - Restart Begin	com.oraclecloud.databaseservice.restartautonomousdatabase.begin
Autonomous AI Database - Restart End	com.oraclecloud.databaseservice.restartautonomousdatabase.end
Autonomous AI Database - Restore Begin	com.oraclecloud.databaseservice.autonomous.database.restore.begin
Autonomous AI Database - Restore End	com.oraclecloud.databaseservice.autonomous.database.restore.end
Autonomous AI Database - Rotate Encryption Key Begin	com.oraclecloud.databaseservice.rotateautonomousdatabaseencryptionkey.begin
Autonomous AI Database - Rotate Encryption Key End	com.oraclecloud.databaseservice.rotateautonomousdatabaseencryptionkey.end
Autonomous AI Database - Start Begin	com.oraclecloud.databaseservice.startautonomousdatabase.begin
Autonomous AI Database - Start End	com.oraclecloud.databaseservice.startautonomousdatabase.end
Autonomous AI Database - Stop Begin	com.oraclecloud.databaseservice.stopautonomousdatabase.begin
Autonomous AI Database - Stop End	com.oraclecloud.databaseservice.stopautonomousdatabase.end
Autonomous AI Database - Terminate Begin	com.oraclecloud.databaseservice.deleteautonomousdatabase.begin
Autonomous AI Database - Terminate End	com.oraclecloud.databaseservice.deleteautonomousdatabase.end
Autonomous AI Database - Update Begin	com.oraclecloud.databaseservice.updateautonomousdatabase.begin
Autonomous AI Database - Update End	com.oraclecloud.databaseservice.updateautonomousdatabase.end
Autonomous AI Database - Warning See Autonomous AI Database Critical, Warning, and Information Event Types for details.	com.oraclecloud.databaseservice.autonomous.database.warning

Example 7-7 Autonomous AI Database Event

This is a reference event for an Autonomous AI Database:

```
{
  "cloudEventsVersion": "0.1",
  "eventID": "<unique_ID>",
  "eventType":
```

```

"com.oraclecloud.databaseservice.autonomous.database.backup.begin",
  "source": "databaseservice",
  "eventTypeVersion": "2.0",
  "eventTime": "2019-07-10T14:06:23Z",
  "contentType": "application/json",
  "extensions": {
    "compartmentId": "ocid1.compartment.oc1..<unique_ID>"
  },
  "data": {
    "compartmentId": "ocid1.compartment.oc1..<unique_ID>",
    "compartmentName": "example_name",
    "resourceName": "my_database",
    "resourceId": "ocid1.autonomousdatabase.oc1.phx.<unique_ID>",
    "availabilityDomain": "SoSC:PHX-AD-3",
    "freeFormTags": {},
    "definedTags": {},
    "additionalDetails": {
      "cpuCoreCount": 1,
      "lifecycleState": "PROVISIONING",
      "dataStorageSizeInTBs": 1,
      "timeCreated": "2019-07-10T14:06:10.905Z",
      "timeUpdated": "2019-07-10T14:06:10.905Z",
      "serviceConsoleUrl": null,
      "licenseType": null,
      "workloadType": "<Lakehouse | Transaction Processing>",
      "autonomousDatabaseType": "<Dedicated Infrastructure | Serverless Infrastructure>"
    }
  }
}

```

Autonomous AI Database Critical, Warning, and Information Event Types

Autonomous AI Databases emit critical, warning, and information event types. These event types allow you to receive notifications:

- When your database needs urgent attention (critical events)
- When an event is not critical but needs your attention to prevent it from turning critical (warning events)
- For an event that is good to be notified for your information (information events)

The critical, warning, and information event types use an `additionalDetails` section of the event message to provide specific details about what is happening within the Autonomous AI Database emitting the event. The `eventType` values for each of these events are listed below:

Friendly Name	Event Type
Autonomous AI Database - Critical	<code>com.oraclecloud.databaseservice.autonomous.database.critical</code>
Autonomous AI Database - Warning	<code>com.oraclecloud.databaseservice.autonomous.database.warning</code>
Autonomous AI Database - Information	<code>com.oraclecloud.databaseservice.autonomous.database.information</code>

In the following example of a critical event, you can see within the `additionalDetails` section of the event message that this particular message concerns an "ADMIN password expired" condition that has just ended. The `eventName` and `description` fields provide information regarding the critical situation within the "adwfinance" database:

```
{
  "eventType": "com.oraclecloud.databaseservice.autonomous.database.critical",
  "cloudEventsVersion": "0.1",
  "eventTypeVersion": "2.0",
  "source": "databaseservice",
  "eventTime": "2021-12-31T12:00:38Z",
  "contentType": "application/json",
  "data": {
    "compartmentId": "<compartment id>",
    "compartmentName": "DevOpsTest",
    "resourceName": "NONDGATP",
    "resourceId": "<resource id>",
    "availabilityDomain": "Efnw:PHX-AD-1",
    "additionalDetails": {
      "systemId": "<cloud autonomous VM cluster OCID>",
      "databaseName": "adwfinance",
      "infrastructureType": "exadata",
      "description": "ADMIN user password has expired and must be reset",
      "eventName": "adminpassword.expired",
      "autonomousDatabaseType": "Dedicated Infrastructure",
      "workloadType": "Transaction Processing"
    }
  },
  "eventID": "26257ab6-5872-42d0-b5b5-cc8b2b82f0f1",
  "extensions": {
    "compartmentId": "<compartment ocid>"
  }
}
```

In the tables below, you can read about the conditions and operations that trigger critical, warning, and information events. Each condition or operation is identified by a unique `eventName` value.

Critical Events for Autonomous AI Database

Event	Description
<code>adminpassword.expired</code>	If the ADMIN password expires and is no longer usable, Autonomous AI Database reports an event that specifies that the ADMIN user password has expired and must be reset.
<code>autonomous.operations.issue.detected.critical</code>	Autonomous Dedicated Operations has detected a critical issue with your Autonomous AI Database and is working on it. No action is required on your part.
<code>autonomous.operations.issue.resolved.critical</code>	Autonomous Dedicated Operations has resolved a critical issue with your Autonomous AI Database. No further action is required.
<code>wallet.expired</code>	Your Autonomous AI Database wallet has expired.
<code>wallet.rotated</code>	Your Autonomous AI Database wallet has been rotated and must be downloaded.

Event	Description
databasevault.disabled	Database Vault for this Autonomous AI Database was disabled by a process other than standard Oracle patching automation. Please review your database security configuration and audit log
databasevault.enabled	Database Vault for this Autonomous AI Database was enabled by a process other than standard Oracle patching automation. Please review your database security configuration and audit logs.
dbservice.status.critical	Service : <service_name> is offline. The Autonomous AI Database operations team is investigating the issue and will provide more information as needed.

Warning Events for Autonomous AI Database

Event	Description
adminpasswordexpiry.warning	Generated when Autonomous AI Database determines that the ADMIN password will expire within 4 weeks. You receive a warning indicating the date when the ADMIN password can no longer be used. If the password is not reset, the event is reported at most once per week. In last week daily event is sent.
autonomous.operations.issue.detected.warning	Autonomous Dedicated Operations has detected an issue with your Autonomous AI Database and is working on it. No action is required on your part.
autonomous.operations.issue.resolved.warning	Autonomous Dedicated Operations has resolved an issue with your Autonomous AI Database. No further action is required.
storagequotalimit.warning	Generated when the storage quota usage exceeds 90%.
walletexpiry.warning	Generated when Autonomous AI Database determines that a wallet is due to expire in less than six (6) weeks. If the wallet is not refreshed, the event is reported at most once per week. In last week daily event is sent.

Information Events for Autonomous AI Database

Event	Description
autonomous.operations.issue.detected.information	Autonomous Dedicated Operations has detected an issue with your Autonomous AI Database and is working on it. No action is required on your part.
autonomous.operations.issue.resolved.information	Autonomous Dedicated Operations has resolved an issue with your Autonomous AI Database. No further action is required.
dbservice.status.information	Service : <service name> is now online.

Availability Service Level Agreements (SLAs)

This topic describes the Service Level Agreements (SLAs) and Service Level Objectives (SLOs) for Oracle Autonomous AI Database on Dedicated Exadata Infrastructure.

Oracle Autonomous AI Database runs on the Oracle Exadata Cloud infrastructure (Oracle Public Cloud, Multicloud, and Oracle Exadata Cloud@Customer), leveraging Oracle's Maximum Availability Architecture (MAA). Autonomous AI Database on Dedicated Exadata Infrastructure is engineered to return an application online following an unplanned outage or a planned maintenance activity within single-digit seconds.

Oracle *Maximum Availability Architecture (MAA)* is a set of best practices developed by Oracle engineers over many years for the integrated use of Oracle High Availability, data protection, and disaster recovery technologies. The key goal of Oracle MAA is to meet Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) for Oracle databases and applications running on our system and database platforms using Oracle Cloud MAA architectures and solutions. Autonomous AI Database on Dedicated Exadata Infrastructure has been MAA Platinum validated and certified. See *Maximum Availability Architecture and Autonomous AI Database Cloud* in Oracle Database 19c High Availability Overview and Best Practices or [Oracle Database 26ai High Availability Overview and Best Practices](#) for more details about Oracle MAA.

Uptime

The following table outlines the Service Level Agreement (SLA) and Service Level Objective (SLO) for Oracle Autonomous AI Database on Dedicated Exadata Infrastructure.

Table 7-22 Uptime SLAs/SLOs

Service	Type	Uptime (without Autonomous Data Guard)	Uptime (with Autonomous Data Guard)
Autonomous AI Database on Dedicated Exadata Infrastructure (Oracle Public Cloud deployments), Autonomous AI Database on Oracle Database@AWS	Service Level Agreement (SLA)	99.95% A maximum of 22 minutes of downtime per month.	99.995% A maximum of 132 seconds of downtime per month.
Autonomous AI Database on Exadata Cloud@Customer	Service Level Objective (SLO)	99.95% A maximum of 22 minutes of downtime per month.	99.995% A maximum of 132 seconds of downtime per month.
Autonomous AI Database for Developers (Both Oracle Public Cloud and Exadata Cloud@Customer deployments)	Service Level Objective (SLO)	99.5%	Not applicable Autonomous AI Database for Developers is not supported with Autonomous Data Guard.

Note

For the Availability Service Level Agreement (SLA) under the Uptime columns in the above table, Oracle will use commercially reasonable efforts to have each such Service available with the indicated Monthly Uptime Percentage during any calendar month (the “Service Commitment”). If this Service Commitment is not met, you will be eligible to receive **Service Credits** for such Non-Compliant Service, with a Service Credit Percentage. Refer to [Oracle PaaS and IaaS Public Cloud Services Pillar Document](#) for the Service Credit Percentage values and other details.

Recovery Time Objective (RTO) and Recovery Point Objective (RPO)

The following tables outline the target Recovery Time Objective (RTO) and Recovery Point Objective (RPO) SLAs/SLOs for different failure events for Autonomous AI Database on Dedicated Exadata Infrastructure without Autonomous Data Guard and with Autonomous Data Guard.

Table 7-23 Default High Availability Policy Recovery Time and Recovery Point SLAs/SLOs

Failure and Maintenance Events	Service-Level Downtime (SLO)	Maximum Viable Data Loss
Localized events, including: - Exadata cluster network topology failures - Storage (disk and flash) failures - Database instance failures - Database server failures - Periodic software and hardware maintenance updates	Near-zero	Zero
Events that require restoring from backup because the standby database does not exist: - Data corruptions - Full database failures - Complete storage failures - Availability domain (AD) for multi-AD regions	Minutes to hours (without Autonomous Data Guard)	15 minutes (without Autonomous Data Guard)
Events that require non-rolling software updates or database upgrades	Until the non-rolling software update or database upgrade event completes. For the upgrades that include a time-zone file update, the service-level downtime depends on the amount of time-zone data that is modified during the upgrade.	Zero

Table 7-24 Autonomous Data Guard Recovery Time and Recovery Point SLAs/SLOs

Failure and Maintenance Events	Service-level Downtime (RTO)	Potential Service-level Data Loss (RPO)
Localized events, including: - Exadata cluster network fabric failures - Storage (disk and flash) failures - Database instance failures - Database server failures - Periodic software and hardware maintenance updates	Zero or Near Zero	Zero
Events that require failover to the standby database using Autonomous Data Guard, including: - Data corruptions (because Data Guard has automatic block repair for physical corruptions, a failover operation is required only for logical corruptions or extensive data corruptions) - Full database failures - Complete storage failures - Availability domain or region failures (Regional failure protection is only available if the standby is located across regions.)	Few seconds to two minutes	- Zero with maximum availability protection mode (uses synchronous redo transport). Most commonly used for intra-region standby databases. - Near zero for maximum performance protection mode (uses asynchronous redo transport). Most commonly used for cross-region standby databases.

API Reference for Autonomous AI Database on Dedicated Exadata Infrastructure

You can use API to view and manage the different infrastructure resources of an Autonomous AI Database on Dedicated Exadata Infrastructure.

For information about using the API and signing requests, see [REST APIs](#) and [Security Credentials](#). For information about SDKs, see [Software Development Kits and Command Line Interface](#).

API to Manage Exadata Infrastructure Resources

The following table lists the REST API endpoints to manage Exadata Infrastructure resources on Oracle Public Cloud and Multicloud.

Operation	REST API Endpoint	Required IAM Permissions
Create an Exadata Infrastructure resource	CreateCloudExadataInfrastructure	CLOUD_EXADATA_INFRASTRUCTURE_CREATE VNIC_CREATE SUBNET_ATTACH VNIC_ATTACH
View a list of Exadata Infrastructure resources	ListCloudExadataInfrastructures	CLOUD_EXADATA_INFRASTRUCTURE_INSPECT
View details of an Exadata Infrastructure resource	GetCloudExadataInfrastructure	CLOUD_EXADATA_INFRASTRUCTURE_INSPECT
Update details an Exadata Infrastructure resource. You can use this API to update compute, storage, contacts, tags, and maintenance window.	UpdateCloudExadataInfrastructure	CLOUD_EXADATA_INFRASTRUCTURE_UPDATE
Move an Exadata Infrastructure resource to another compartment	ChangeCloudExadataInfrastructureCompartment	CLOUD_EXADATA_INFRASTRUCTURE_UPDATE
Terminate an Exadata Infrastructure resource	DeleteCloudExadataInfrastructure	CLOUD_EXADATA_INFRASTRUCTURE_DELETE VNIC_DELETE SUBNET_DETACH VNIC_DETACH

The following table lists the REST API endpoints to manage Exadata Infrastructure resources on Exadata Cloud@Customer.

Operation	REST API Endpoint	Required IAM Permissions
Create an Exadata Infrastructure resource	CreateExadataInfrastructure	EXADATA_INFRASTRUCTURE_CREATE
View a list of Exadata Infrastructure resources	ListExadataInfrastructures	EXADATA_INFRASTRUCTURE_INSPECT
View details of an Exadata Infrastructure resource	GetExadataInfrastructure	EXADATA_INFRASTRUCTURE_INSPECT
Update details an Exadata Infrastructure resource. You can use this API to update compute, storage, contacts, tags, and maintenance window.	UpdateExadataInfrastructure	EXADATA_INFRASTRUCTURE_UPDATE
Move an Exadata Infrastructure resource to another compartment	ChangeExadataInfrastructureCompartment	EXADATA_INFRASTRUCTURE_INSPECT and EXADATA_INFRASTRUCTURE_UPDATE
Terminate an Exadata Infrastructure resource	DeleteExadataInfrastructure	EXADATA_INFRASTRUCTURE_DELETE
Download an Exadata Infrastructure resource configuration file	DownloadExadataInfrastructureConfigFile	EXADATA_INFRASTRUCTURE_CONTENT_READ

Operation	REST API Endpoint	Required IAM Permissions
Add storage capacity to an Exadata Infrastructure	AddStorageCapacityExadataInfrastructure	EXADATA_INFRAStructure_UPDATE
Activate an Exadata Infrastructure	ActivateExadataInfrastructure	EXADATA_INFRAStructure_UPDATE

API to Manage Autonomous Exadata VM Clusters

The following table lists the REST API endpoints to manage Autonomous Exadata VM Clusters on Oracle Public Cloud and Multicloud.

Operation	REST API Endpoint	Required IAM Permissions
Create an Autonomous Exadata VM Cluster	CreateCloudAutonomousVmCluster	CLOUD_AUTONOMOUS_VM_CLUSTER_CREATE and CLOUD_EXADATA_INFRAStructure_UPDATE
View details of an Autonomous Exadata VM Cluster	GetCloudAutonomousVmCluster	CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT
View a list of Autonomous Exadata VM Clusters	ListCloudAutonomousVmClusters	CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT
Move an Autonomous Exadata VM Cluster to another compartment	ChangeCloudAutonomousVmClusterCompartment	CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT and CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE
Terminate an Autonomous Exadata VM Cluster	DeleteCloudAutonomousVmCluster	CLOUD_AUTONOMOUS_VM_CLUSTER_DELETE
Rotate ORDS Certificates	RotateCloudAutonomousVmClusterOrdsCerts	CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE
Rotate SSL Certificates	RotateCloudAutonomousVmClusterSslCerts	CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE
Update details of Autonomous Exadata VM Clusters. You can modify the tags, license model, or Network Security Groups (NSG) using this API.	UpdateCloudAutonomousVmCluster	CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE
Get the resource usage details of an Autonomous Exadata VM Cluster and the usage statistics of the Autonomous Container Databases within it.	GetAutonomousVMClusterResourceUsage	CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT

The following table lists the REST API endpoints to manage Autonomous Exadata VM Clusters on Exadata Cloud@Customer.

Operation	REST API Endpoint	Required IAM Permissions
Create an Autonomous Exadata VM Cluster	CreateAutonomousVmCluster	AUTONOMOUS_VM_CLUSTER_CREATE and EXADATA_INFRAStructure_INSPECT and EXADATA_INFRAStructure_UPDATE

Operation	REST API Endpoint	Required IAM Permissions
View a list of Autonomous Exadata VM Clusters	ListAutonomousVmClusters	AUTONOMOUS_VM_CLUSTER_INSPECT
View details of an Autonomous Exadata VM Cluster	GetAutonomousVmCluster	AUTONOMOUS_VM_CLUSTER_INSPECT
Change the license type of an Autonomous Exadata VM Cluster	UpdateAutonomousVmCluster	AUTONOMOUS_VM_CLUSTER_UPDATE and EXADATA_INFRASTRUCTURE_INSPECT and EXADATA_INFRASTRUCTURE_UPDATE
Move an Autonomous Exadata VM Cluster to another compartment	ChangeAutonomousVmClusterCompartment	AUTONOMOUS_VM_CLUSTER_INSPECT and AUTONOMOUS_VM_CLUSTER_UPDATE
Terminate an Autonomous Exadata VM Cluster	DeleteAutonomousVmCluster	AUTONOMOUS_VM_CLUSTER_DELETE

API to Manage Autonomous Container Databases

The following table lists the REST API endpoints to manage Autonomous Container Databases.

Operation	REST API Endpoint	Required IAM Permissions
Create an Autonomous Container Database	CreateAutonomousContainerDatabase	On Oracle Public Cloud and Multicloud: CLOUD_EXADATA_INFRASTRUCTURE_UPDATE and CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_CREATE On Exadata Cloud@Customer: EXADATA_INFRASTRUCTURE_UPDATE and AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_CREATE
View a list of Autonomous Container Databases	ListAutonomousContainerDatabases	AUTONOMOUS_CONTAINER_DATABASE_INSPECT
View details of an Autonomous Container Database	GetAutonomousContainerDatabase	AUTONOMOUS_CONTAINER_DATABASE_INSPECT
Change the backup retention policy of an Autonomous Container Database or Edit the maintenance preferences of an Autonomous Container Database	UpdateAutonomousContainerDatabase	AUTONOMOUS_CONTAINER_DATABASE_UPDATE
Restart an Autonomous Container Database	RestartAutonomousContainerDatabase	AUTONOMOUS_CONTAINER_DATABASE_UPDATE

Operation	REST API Endpoint	Required IAM Permissions
Move an Autonomous Container Database to another compartment	ChangeAutonomousContainerDatabaseCompartment	AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE
Rotate an Autonomous Container Database key	RotateAutonomousContainerDatabaseEncryptionKey	AUTONOMOUS_CONTAINER_DATABASE_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_INSPECT
Terminate an Autonomous Container Database	TerminateAutonomousContainerDatabase	On Oracle Public Cloud and Multicloud: CLOUD_EXADATA_INFRASTRUCTURE_UPDATE and CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_DELETE On Exadata Cloud@Customer: EXADATA_INFRASTRUCTURE_UPDATE and AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_DELETE
Get the resource usage statistics of an Autonomous Container Database	GetAutonomousContainerDatabaseResourceUsage	AUTONOMOUS_CONTAINER_DATABASE_INSPECT
View a list of Autonomous Container Database backups	ListAutonomousContainerDatabaseBackups	AUTONOMOUS_CONTAINER_DATABASE_BACKUP_INSPECT

API to Manage Autonomous AI Databases

The following table lists the REST API endpoints to manage Autonomous AI Databases.

Operation	REST API Endpoint	Required IAM Permissions
Create an Autonomous AI Database or Clone an Autonomous AI Database	CreateAutonomousDatabase	AUTONOMOUS_DATABASE_CREATE
View a list of Autonomous AI Databases	ListAutonomousDatabases	AUTONOMOUS_DATABASE_INSPECT
View details of an Autonomous AI Database	GetAutonomousDatabase	AUTONOMOUS_DATABASE_INSPECT
View a list of character sets supported by Autonomous AI Database.	ListAutonomousDatabaseCharacterSets	None

Operation	REST API Endpoint	Required IAM Permissions
Set the password of an Autonomous AI Database's ADMIN user or Scale the CPU core count or storage of an Autonomous AI Database or Enable or disable auto scaling for an Autonomous AI Database	UpdateAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
Move an Autonomous AI Database to another compartment	ChangeAutonomousDatabaseCompartment	AUTONOMOUS_DATABASE_UPDATE and AUTONOMOUS_DB_BACKUP_INSPECT and AUTONOMOUS_DB_BACKUP_CONTENT_READ and AUTONOMOUS_DATABASE_CONTENT_WRITE
Rotate an Autonomous AI Database key	RotateAutonomousDatabaseEncryptionKey	AUTONOMOUS_DATABASE_UPDATE
Start an Autonomous AI Database	StartAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
Stop an Autonomous AI Database	StopAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
Restart an Autonomous AI Database	RestartAutonomousDatabase	AUTONOMOUS_DATABASE_UPDATE
Terminate an Autonomous AI Database	DeleteAutonomousDatabase	AUTONOMOUS_DATABASE_DELETE
Restore an Autonomous AI Database	RestoreAutonomousDatabase	AUTONOMOUS_DB_BACKUP_CONTENT_READ and AUTONOMOUS_DATABASE_CONTENT_WRITE
Back up an Autonomous AI Database manually	CreateAutonomousDatabaseBackup	AUTONOMOUS_DB_BACKUP_CREATE and AUTONOMOUS_DATABASE_CONTENT_READ
View the list of Autonomous AI Database backups	ListAutonomousDatabaseBackups	AUTONOMOUS_DB_BACKUP_INSPECT
Get the details of an Autonomous AI Database backup	GetAutonomousDatabaseBackup	AUTONOMOUS_DB_BACKUP_INSPECT
Delete a long-term backup	DeleteAutonomousDatabaseBackup	AUTONOMOUS_DB_BACKUP_DELETE
Update the retention period of a long-term backup	UpdateAutonomousDatabaseBackup	AUTONOMOUS_DB_BACKUP_UPDATE

API to Manage Autonomous Data Guard Configuration

The following table lists the REST API endpoints to manage an Autonomous Data Guard configuration.

Operation	API Endpoint	Required IAM Permissions
View the Autonomous Data Guard Association with an ACD.	GetAutonomousContainerDatabase	AUTONOMOUS_CONTAINER_DATABASE_INSPECT
List the ACDs enabled with Autonomous Data Guard associated with the specified ACD or Autonomous AI Database.	ListAutonomousContainerDatabases	AUTONOMOUS_CONTAINER_DATABASE_INSPECT
Reinstate the Disabled Standby to an active standby ACD.	ReinstateAutonomousContainerDatabaseDataguard	AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE
Switch Roles of the primary and standby ACDs.	SwitchoverAutonomousContainerDatabaseDataguard	AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE
Fail Over to the Standby ACD. This standby ACD will become the new primary ACD when the failover completes successfully.	FailoverAutonomousContainerDatabaseDataguard	AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE
Modify Autonomous Data Guard settings such as protection mode, automatic failover, and fast start failover lag limit.	EditAutonomousContainerDatabaseDataguard	AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE
Get an Autonomous Data Guard-enabled database associated with the specified Autonomous AI Database.	GetAutonomousContainerDatabase	AUTONOMOUS_CONTAINER_DATABASE_INSPECT
List Autonomous AI Database data guard associations.	GetAutonomousContainerDatabase	AUTONOMOUS_CONTAINER_DATABASE_INSPECT
Enable Autonomous Data Guard on an ACD.	AddStandbyAutonomousContainerDatabase	CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT, AUTONOMOUS_CONTAINER_DATABASE_INSPECT, and AUTONOMOUS_CONTAINER_DATABASE_UPDATE
Convert the standby ACD between physical standby and snapshot standby ACD.	ConvertStandbyAutonomousContainerDatabase	CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT, AUTONOMOUS_CONTAINER_DATABASE_INSPECT, and AUTONOMOUS_CONTAINER_DATABASE_UPDATE

API to Manage Autonomous Virtual Machines

The following table lists the REST API endpoints to manage an Autonomous Virtual Machines.

Operation	REST API Endpoint	Required IAM Permissions
Gets the details of a specific Autonomous Virtual Machine.	GetAutonomousVirtualMachine	AUTONOMOUS_VIRTUAL_MACHINE_INSPECT

Operation	REST API Endpoint	Required IAM Permissions
Lists the Autonomous Virtual Machines in the specified Autonomous VM Cluster and Compartment.	ListAutonomousVirtualMachines	AUTONOMOUS_VIRTUAL_MACHINE_INSPECT

API to Manage Autonomous AI Database Software Images

The following table lists the REST API endpoints to manage an Autonomous AI Database software image.

Operation	REST API Endpoint	Required IAM Permissions
Create an Autonomous AI Database software image in the specified compartment.	CreateAutonomousDatabaseSoftwareImage	AUTO_DB_SOFTWARE_IMAGE_CREATE
Move an Autonomous AI Database software image and its dependent resources to the specified compartment.	ChangeAutonomousDatabaseSoftwareImageCompartment	AUTO_DB_SOFTWARE_IMAGE_UPDATE
Delete an Autonomous AI Database software image.	DeleteAutonomousDatabaseSoftwareImage	AUTO_DB_SOFTWARE_IMAGE_DELETE
Get information about the specified Autonomous AI Database software image.	GetAutonomousDatabaseSoftwareImage	AUTO_DB_SOFTWARE_IMAGE_INSPECT
Get a list of the Autonomous AI Database software images in the specified compartment.	ListAutonomousDatabaseSoftwareImages	AUTO_DB_SOFTWARE_IMAGE_INSPECT
Update details of an Autonomous AI Database software image.	UpdateAutonomousDatabaseSoftwareImage	AUTO_DB_SOFTWARE_IMAGE_UPDATE

Troubleshooting Autonomous AI Database on Dedicated Exadata Infrastructure

Use the following sections to help troubleshoot problems you have with Oracle Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Public Cloud and Exadata Cloud@Customer platforms.

Unable to Access Master Encryption Keys

APPLIES TO:  Oracle Public Cloud only

Potential Cause

Autonomous Exadata VM Cluster (AVMC) is unable to reach the master encryption key.

Suggested Action

Verify that the master encryption key is reachable from the AVMC.

Resolution

- Ensure Service Gateway is enabled for the AVMC Subnet with *Destination: All IAD Services In Oracle Services Network*.
- Ensure the following IAM policy is defined for the Dynamic Group:

```
allow dynamic-group <dynamic-group-name>
to manage keys
in compartment <vaults-and-keys-compartment>
where all {
    target.key.id='<key_ocid>',
    request.permission!='KEY_DELETE',
    request.permission!='KEY_MOVE',
    request.permission!='KEY_IMPORT',
    request.permission!='KEY_BACKUP'
}
```

Further Reference

- Live Lab: [Oracle Autonomous AI Database Dedicated for Security Administrators](#)
- Documentation: Prepare to Use Customer-Managed Keys in the Vault Service

Unable to Access Vault

APPLIES TO:  Oracle Public Cloud only

Potential Cause

Autonomous Exadata VM Cluster (AVMC) is unable to read the Vault.

Suggested Action

Verify that the Vault is reachable from the AVMC.

Resolution

- Ensure Service Gateway is enabled for the AVMC Subnet with *Destination: All IAD Services In Oracle Services Network*
- Ensure the following IAM policy is defined for the Dynamic Group

```
allow dynamic-group <dynamic-group>
to read vaults
in tenancy | compartment <vaults-and-keys-compartment>
```

Further Reference

- Live Lab: [Oracle Autonomous AI Database Dedicated for Security Administrators](#)
- Documentation: Prepare to Use Customer-Managed Keys in the Vault Service

Backup to Network File System (NFS) is Failing

APPLIES TO: • Exadata Cloud@Customer only

Potential Cause

NFS destination may be unreachable due to a network issue.

Suggested Action

Verify that the NFS is reachable from the Autonomous Exadata VM Cluster (AVMC) network.

Resolution

- Verify the network routing and try again. All IP Addresses need to be reachable over backup network of the AVMC.
- Disconnect and re-attach the NFS.
- Attach Secondary NFS shared.

Further Reference

Documentation: About Backup and Recovery

Unable to Mount Network File System (NFS)

APPLIES TO: • Exadata Cloud@Customer only

Potential Causes

- Incorrect export path.
- Lack of right permissions on the export path.
- No network access between Autonomous Exadata VM Cluster (AVMC) client IPs and NFS server.

Suggested Actions

- Verify the export path and permissions on the export path.
- Verify access ports are open between NFS server and client IPs.

Resolution

- Ensure the **export_path** is accurate.
- Ensure **Oracle** user has permission on the **export_path**
 - **uid:gid** of the Oracle user for Autonomous VM Cluster should be **1001:1001**
- Ensure no firewalls block network access between AVMC client IPs and the NFS server.
- If the network access to NFS is via backup IPs, then create an SR for the Autonomous AI Database operations to implement routing rules to divert traffic to NFS via backup IPs.

Unable to write a file in Network File System (NFS)

APPLIES TO: • Exadata Cloud@Customer only

Potential Cause

Incorrect permissions on the NFS mount.

Suggested Action

Verify if the NFS mount has the right permission.

Resolution

Ensure Oracle user has the right permission on the NFS mount.

- **uid:gid** of the Oracle user for Autonomous VM Cluster should be **1001:1001**

Unable to get outbound traffic from APEX

Error Code

OPC :ORA-24247 WHILE TRYING TO USE APEX_INSTANCE_ADMIN.VALIDATE_EMAIL_CONFIG

Potential Cause

Missing https & SMTP egress rules.

Suggested Action

Enable network access for APEX as per your requirement for tasks such as sending email, or accessing REST (or other HTTP based) resources. Access will only be available after the user configures it.

Resolution

To grant network access through APEX (as was previously configured out of the box):

- The principal name specified must match the APEX installation schema, for example it might be APEX_210200.
- The apex schema name for a particular deployment is version dependent and can be found with the following query:

```
select schema from dba_registry where comp_id='APEX'
```

- ACLs created for other users, such as ADMIN, will not impact access through APEX. Such ACLs would only impact use cases where ADMIN or code owned by ADMIN directly called UTL_HTTP or UTL_SMTP.

Further Reference

- Documentation that demonstrates the wildcard rule that would allow access to any hosts, and a subsequent more restrictive rule that allowed localhost access: Enabling Network Services in Oracle Database

- More details about adding rules to allow specific hosts or wildcard patterns can be found in *APPEND_HOST_Procedure* in Oracle Database 19c PL/SQL Packages and Types Reference or [Oracle Database 26ai PL/SQL Packages and Types Reference](#).

Backup Retention Issues with ZDLRA

APPLIES TO: • Exadata Cloud@Customer only

Potential Cause

The root cause depends on the details in the diagnostic report generated per the suggestions below.

Suggested Action

- To help us understand and fix the issue, please follow the following My Oracle Support (MOS) note that explains you through collecting diagnostic data, which will help Support narrow down the cause of your problem.
[System Activity Report: SRDC - Zero Data Loss Recovery Appliance \(ZDLRA\) Data Collection \(Doc ID 2154189.1\)](#)
- If the Recovery Appliance is higher than or equal to Version 19.2.1.1.2, the following command can also be used to generate the System Activity Report (SAR) but this will be generated in **text format** only:

```
racli run diagnostics --tag=sar
```

This command generates a Diagnostics package.

- Submit a Service Request (SR) in My Oracle Support with the diagnostic results.

Resolution

Once you submit the SR with the diagnostic results, the Oracle Support team will contact you with a resolution for the corresponding issue.

Further Reference

- Documentation: Create a Service Request in My Oracle Support

Delay in Connecting to Autonomous AI Database with SQL*Plus

Potential Cause

Whenever you try to connect to Autonomous AI Database on Dedicated Exadata Infrastructure with SQL*Plus, it attempts to reach ONS on port 6200 to subscribe to FAN events, once per node. If the request to reach node 6200 expires due to a timeout after 10 seconds (per node), it proceeds with regular SQLNET 1521/2484, resulting in a delay in connection.

You may have a blocked ingress on the domU for port 6200, or a blocked egress on the client host.

Suggested Action

Provide ingress and egress rules for FAN port 6200 or disable subscription to FAN events on your client.

Resolution

You have two options to resolve this issue:

- Disable subscription to FAN events on your client by updating the oraaccess XML file.

The oraaccess.xml file allows you to override the connection-specific parameters based on the individual application requirements. See [Overriding Connection Parameters at the Connection-String Level](#) for examples.

- Fix connectivity to port 6200 from the client host to domU by providing the ingress and egress rules for FAN port 6200 as described in Step 4. Create the VCN and Subnets.

Further Reference

- [Overriding Connection Parameters at the Connection-String Level](#)
- [Step 4. Create the VCN and Subnets](#)

Certify Applications for Compatibility Between Autonomous AI Database Serverless and Dedicated

Considering that your application is already certified to use Autonomous AI Database Serverless, the following table helps you evaluate if your application can be certified to use Autonomous AI Database on Dedicated Exadata Infrastructure too.

The table below lists the options and features that are either:

- Currently available only in Autonomous AI Database Serverless, or
- Has *differences in functionality* with Autonomous AI Database on Dedicated Exadata Infrastructure deployments both in Public Cloud and Cloud@Customer.

Note

There are also features only supported in Autonomous AI Database on Dedicated Exadata Infrastructure deployment and not supported in Autonomous AI Database Serverless deployments. However, as the presumption is that you are evaluating to certify your application on both the deployments, starting with Autonomous AI Database Serverless deployments, we are not discussing them here.

Table 7-25 Autonomous AI Database Serverless Vs Autonomous AI Database on Dedicated Exadata Infrastructure Features

Feature/Option	Autonomous AI Database Serverless	Autonomous AI Database on Dedicated Exadata Infrastructure
Database Renaming	Supported.	Not supported currently.
Database Resident Connection Pooling	Use Database Resident Connection Pooling with Autonomous AI Database Serverless	Not supported currently.

Table 7-25 (Cont.) Autonomous AI Database Serverless Vs Autonomous AI Database on Dedicated Exadata Infrastructure Features

Feature/Option	Autonomous AI Database Serverless	Autonomous AI Database on Dedicated Exadata Infrastructure
DBMS_AQADM	Supported.	Supported with the following limitations: - DBMS_AQADM.MOVE_QUEUE_TABLE can only be used in ONLINE mode to move Advanced Queuing (AQ) tables from one tablespace to another or within the same tablespace. - DBMS_AQADM.MOVE_QUEUE_TABLE can not be used to move AQ tables even in ONLINE mode when table or schema names are case-sensitive and include special identifiers.
DBMS_CLOUD_MACADM	Supported.	Not supported currently.
DBMS_MAX_STRING_SIZE	Supported.	Not supported currently.
Graph Studio	Supported.	Not supported currently.
Modifiable Initialization Parameters	MAX_STRING_SIZE See List of Initialization Parameters that can be Modified in Autonomous AI Database Serverless.	For differences with Autonomous AI Database Serverless, see Modifying Database Initialization Parameters.
Oracle Application Express (APEX) Support	See Restrictions and Limitations for Oracle Application Express with Autonomous AI Database.	For differences with Autonomous AI Database Serverless, see Limitations on the Use of Oracle Application Express.
Oracle Cloud Infrastructure PL/SQL SDK API	Supported.	Not supported currently.
Oracle Spatial and Graph	This feature is provided through Oracle Maps Cloud Service. - Geocoder - Routing Engine - Spatial Visualization See Oracle Spatial Limitations with Autonomous AI Database Serverless.	For differences with Autonomous AI Database Serverless, see Oracle Cloud Marketplace Images for Spatial.

Table 7-25 (Cont.) Autonomous AI Database Serverless Vs Autonomous AI Database on Dedicated Exadata Infrastructure Features

Feature/Option	Autonomous AI Database Serverless	Autonomous AI Database on Dedicated Exadata Infrastructure
Oracle Machine Learning (OML)	Supports: • OML4SQL on all database versions • OML4Py • OML4R • OML Notebooks (SQL, PL/SQL, Python, R, markdown) • OML AutoML UI • OML Services • Oracle Data Miner For OML product documentation, refer to Oracle Machine Learning on Oracle Help Center.	Supports: • OML4SQL on all database versions. • Oracle Data Miner For OML product documentation, refer to Oracle Machine Learning on Oracle Help Center.
Read-only, Restricted Database Modes	Supported.	Not supported currently.
Refreshable Clones	Supported.	Not supported currently.
Time-zone file management	Use the DBMS_CLOUD_ADMIN.ENABLE_FEATURE and DBMS_CLOUD_ADMIN.DISABLE_FEATURE procedures to manage time-zone file updates. See Manage Time Zone File Version on Autonomous AI Database for more details.	You can choose to include or exclude a time-zone file update along with the automatic quarterly maintenance updates. See Update Autonomous Container Database Maintenance Preferences for detailed instructions. You can also schedule an on-demand maintenance to update RU (Release Update) along with the time-zone file or just the time-zone file for an ACD. Refer to Schedule a Quarterly Maintenance Update for instructions.
UTL_HTTP	Supported.	Supported. For differences with Autonomous AI Database Serverless, see Notes on the Use of Database PL/SQL Packages.
UTL_SMTP	Supported.	Supported. For differences with Autonomous AI Database Serverless, see Notes on the Use of Database PL/SQL Packages.

Feature Parity Between Oracle Public Cloud and Exadata Cloud@Customer

Autonomous AI Database on Dedicated Exadata Infrastructure offers two deployment possibilities for operational control and isolation; Oracle Public Cloud or Exadata

Cloud@Customer. Even though both platforms support a standard set of features for Autonomous AI Databases, there are a few differences among them.

Feature Comparison

The following table lists the Autonomous AI Database features exclusive to either Oracle Public Cloud or Exadata Cloud@Customer deployments.

Feature	Oracle Public Cloud	Exadata Cloud@Customer
Add a cross tenancy Autonomous Data Guard standby database using Autonomous Data Guard Groups model	✔	-
Change the license type on an Autonomous Exadata VM Cluster (AVMC)	Supported via service request (SR) in My Oracle Support. See Create a Service Request in My Oracle Support for instructions.	✔
Create a cross tenancy clone from a database instance or its backup	✔	-
Configure Zero Trust Packet Routing (ZPR) security attributes for an Autonomous Exadata VM Cluster (AVMC)	✔	-
Connect Python applications using the python-oracledb driver	-	✔
Deploy Autonomous AI Database to X10M Exadata Database Machine Systems	-	✔
Deploy Autonomous AI Database to Exadata Base Machine Systems	✔	On Exadata Cloud@Customer deployments, Base Shape is offered as a fixed shape (Base Rack) while provisioning an Exadata system such as X8M-2, X9M-2 and so on.
Enable cross region backup copy of an Autonomous Container Database	✔	-
Use Database Management Service to monitor databases	✔	-
Use Ops Insights on Autonomous AI Database	✔	-
Bring Your Own Keys (BYOK) on Oracle Public Cloud deployments	✔	-
Use OCI External Key Management System (EKMS) Keys with Oracle Public Cloud deployments.	✔	-

Characteristics of Infrastructure Shapes

This is a reference article for characteristics of Exadata Infrastructure resources for the different Exadata system models and configurations available for Autonomous AI Database on Dedicated Exadata Infrastructure.

Note

See Resource Limits for the maximum and recommended resource limits for Autonomous AI Database on Dedicated Exadata Infrastructure on Oracle Public Cloud and Exadata Cloud@Customer.

Exadata Base Systems

Autonomous AI Database on Oracle Public Cloud

An Exadata base system is a fixed shape similar in size to a quarter rack , with some differences in capacity.

The following table lists the characteristics of the Exadata Base System available in Autonomous AI Database on Dedicated Exadata Infrastructure.

Property	Value
Shape Name	Exadata.Base.48
Number of Compute Nodes	2
Total Minimum Number of Enabled CPU Cores	0
Total Maximum Number of Enabled CPU Cores	48
Total RAM Capacity (GB)	720
Number of Exadata Storage Servers	3
Total Raw Flash Storage Capacity (TB)	38.4
Total Usable Storage Capacity (TB)	74

Note

On Exadata Cloud@Customer deployments, Base Shape is offered as a fixed shape (Base Rack) while provisioning an Exadata system such as X8M-2, X9M-2 and so on.

Exadata X11M Systems

X11M systems support elastic configurations instead of fixed shapes such as quarter rack, half rack, or full rack. X11M systems are currently provisioned with a configuration equivalent to a quarter rack.

Elastic infrastructure shapes range from 2 database and 3 storage servers, up to a total of 32 database and 64 storage servers.

To provision an X11M system successfully, the service limits for Exadata X11M Database Server Count and Exadata X11M Storage Server Count in your tenancy should be at least 2 and 3, respectively. Request a service limit increase from the Oracle Cloud Infrastructure console if required. Observe that the Exadata X11M Database Server Count and Exadata X11M Storage Server Count limits are defined as counts per availability domain. Refer to Request a Service Limit Increase for more details.

Note

Only ECPUs are supported on X11M systems.

Autonomous AI Database on Oracle Public Cloud

The table below lists the characteristics of a Exadata X11M system on Oracle Public Cloud with its default database server and storage server settings, along with several examples of elastic configurations.

Property	Base System	Elastic Configuration Example 1	Elastic Configuration Example 2	Elastic Configuration Example 3
Number of Database (DB) Servers per System	2	2	8	2
Number of Storage Servers per System	3	3	8	14
Total ECPUs in DB Servers per System	192	1520	6080	1520
Minimum Number of ECPUs per VM	20	20	20	20
Total Memory Available for VMs (GB)	720	2780	11120	2780
Maximum Number of VMs per DB Server	1	6	6	6
Maximum Usable Local Storage Per DB Server (GB)	900	2243	2243	2243
Maximum Usable Local Storage Per VM (GB)	900	900	900	900
VM Image Size Minimum and Default (GB)	184	184	184	184
Total Cores in Storage Servers per System	144	192	512	896
Total XRMEM Capacity (TB)	n/a	3.75	10	17.5
Total Flash Capacity (TB)	38.4	81.6	217.6	380.8
Total Usable Disk Capacity (TB)	73	240	640	1120
Maximum DB Size – No Local Backup (TB)	58	192	512	896
Maximum DB Size – Local Backup (TB)	29	96	256	448

Property	Base System	Elastic Configuration Example 1	Elastic Configuration Example 2	Elastic Configuration Example 3
Maximum SQL Flash Bandwidth (GB/s)	25	300	800	1400
Maximum SQL XRMEM Bandwidth (GB/s)	n/a	1500	4000	7000
Maximum SQL Read IOPS	562,500	5,600,000	22,400,000	5,600,000
Maximum SQL Write IOPS	518,000	3,000,000	8,000,000	5,000,000
Maximum SQL Disk Bandwidth (GB/s)	2.7	5.4	14.4	25
Maximum SQL Disk IOPS	3900	7800	20800	36000
Maximum Data Load Rate (TB/hr)	3.8	7.5	20	7.5
Network Connectivity	10 GbE	50 GbE	50 GbE	50 GbE

Individual Server Specifications

Server Type	Total ECPUs	Total Memory Available for VMs (GB)
X11M Database Server (32 Maximum)	760	1390

Server Type	Total Cores	XRMEM Capacity (TB)	Total Flash Capacity (TB)	Total Usable Disk Capacity (TB)
X11M Storage Server (64 Maximum)	64	1.25	27.2	80

For more details, see the [Autonomous AI Database on Dedicated Exadata Infrastructure X11M data sheet](#).

Autonomous AI Database on Exadata Cloud@Customer

The table below lists the characteristics of a Exadata X11M system on Exadata Cloud@Customer with its default database server and storage server settings, along with several examples of elastic configurations.

Property	Base System	Elastic Configuration Example 1	Elastic Configuration Example 2	Elastic Configuration Example 3
Number of Database (DB) Servers per System	2 (Base)	2 (Std, L, XL)	8 (Std, L, XL)	2 (Std, L, XL)

Property	Base System	Elastic Configuration Example 1	Elastic Configuration Example 2	Elastic Configuration Example 3
Number of Storage Servers per System	3 (Base)	3 (HC)	8 (HC)	14 (HC)
Total ECPUs in DB Servers per System	240	1520	6080	1520
Minimum Number of ECPUs per VM	20	20	20	20
Total Memory Available for VMs - Std (GB)	1320	2780	11120	2780
Total Memory Available for VMs - L (GB)	n/a	4180	16720	4180
Total Memory Available for VMs - XL (GB)	n/a	5600	22400	5600
Maximum Number of VMs per DB Server	2	6	6	6
Maximum Usable Local Storage Per DB Server (GB)	1084	2243	2243	2243
Maximum Usable Local Storage Per VM (GB)	900	900	900	900
VM Image Size Minimum and Default (GB)	184	184	184	184
Total Cores in Storage Servers per System	96	192	512	896
Total XRMEM Capacity (TB)	n/a	3.75	10	17.5
Total Flash Capacity (TB)	38.4	81.6	217.6	380.8
Total Usable Disk Capacity (TB)	106	240	640	1120
Maximum DB Size – No Local Backup (TB)	85	192	512	896
Maximum DB Size – Local Backup (TB)	42	96	256	448
Maximum SQL Flash Bandwidth (GB/s)	37.5	300	800	1400
Maximum SQL XRMEM Bandwidth (GB/s)	n/a	1500	4000	7000
Maximum SQL Read IOPS	597,000	5,6000,000	22,400,000	5,600,000

Property	Base System	Elastic Configuration Example 1	Elastic Configuration Example 2	Elastic Configuration Example 3
Maximum SQL Write IOPS	544,000	3,000,000	8,000,000	5,000,000
Maximum SQL Disk Bandwidth (GB/s)	2.7	5.4	14.4	25.0
Maximum SQL Disk IOPS	3900	7800	20800	36000
Maximum Data Load Rate (TB/hr)	3.8	7.5	20.0	7.5

Network Connectivity:

- Per Database Server:
 - 4 x 10/25 Gb SFP28 Ethernet ports (2 client, 2 backup), or
 - 4 x 10 Gb RJ45 Ethernet ports (2 client, 2 backup)
- Per Database Server:Per Control Plane Server:
 - 2 x 10/25 Gb SFP28 Ethernet ports or 2 x 10Gb RJ45 Ethernet ports (Minimum internet connectivity of 50Mbps down and 10Mbps up required)
- Transceiver support for Base Database server:
 - With SFP28 client network, backup network can be RJ45, SFP28, or backup/client on shared SFP28
 - With RJ45 client network, backup network can be SFP28, RJ45, or backup/client on shared RJ45
- Transceiver support for Standard, Large, Extra Large Database Servers:
 - With SFP28 client network, backup network can be SFP28
 - With RJ45 client network, backup network can be RJ45

Individual Server Specifications

Server Type	Total ECPUs	Total Memory Available for VMs (GB)
Base	120	660
Standard (Std)	760	1390
Large (L)	760	2090
Extra Large (XL)	760	2800

Server Type	Total Cores	XRMEM Capacity (TB)	Total Flash Capacity (TB)	Total Usable Disk Capacity (TB)
Base	32	0	12.8	35.6
Standard (Std)	64	1.25	27.2	80

For more details, see the [Autonomous AI Database on Exadata Cloud@Customer X11M data sheet](#).

Exadata X10M Systems

Autonomous AI Database on Exadata Cloud@Customer

For Autonomous AI Databases deployed on Exadata Cloud@Customer, you can select any of the following shapes while provisioning an X10M system.

The following table lists the characteristics of the Exadata X10M Systems available in Autonomous AI Database on Dedicated Exadata Infrastructure deployed on Exadata Cloud@Customer.

Property	Base System	Elastic X10M (min config)	Elastic X10M-L (min config)	Elastic X10M-XL (min config)
Shape Name	ExadataCC.BaseX10M	ExadataCC.X10M	ExadataCC.X10M.L	ExadataCC.X10M.XL
Minimum number of X10M DB servers	2	2	2	2
Minimum number of X10M Storage Servers	3	3	3	3
Total number of usable DB node OCPUs per DB server	30 per server	190 per server	190 per server	190 per server
Total memory available for VMs (GB)	1320	2780	4180	5600
Maximum Local Storage per Database Server (GB)	1084	2243	2243	2243
Max /u02 capacity (GB) per VM	900	900	900	900
VM Image overhead per VM (GB)	184	184	184	184
Max local storage (/u02 + VM Image capacity) (GB) per Infrastructure	1084	2243	2243	2243
Total number of VM clusters*	4	6	6	6
Minimum CPUs per VM	2	2	2	2
Total Usable ASM storage capacity (TB) per Cell	35.6	80	80	80
Exadata RMDA Memory (XRMEM) in TB	0	1.25 per storage server	1.25 per storage server	1.25 per storage server

Exadata X9M-2 Systems

Autonomous AI Database on Oracle Public Cloud

For Autonomous AI Databases deployed on Oracle Cloud, X9M-2 systems support elastic configurations instead of fixed shapes such as quarter rack, half rack, or full rack. X9M-2 systems are currently provisioned with a configuration equivalent to a quarter rack.

Should you need to provision an X9M-2 system with a different configuration, you have the flexibility to choose another value for database servers and storage servers within a valid range. The acceptable range of values is 2 to 32 for database servers and 3 to 64 for storage servers.

To provision an X9M-2 system successfully, the service limits for *Exadata X9M Database Server Count* and *Exadata X9M Storage Server Count* in your tenancy should be at least 2 and 3, respectively. Request a service limit increase from the Oracle Cloud Infrastructure console if required. Observe that the Exadata X9M Database Server Count and Exadata X9M Storage Server Count limits are defined as counts per *availability domain*. Refer to Request a Service Limit Increase for more details.

The table below lists the characteristics of an Exadata X9M-2 system on Oracle Cloud with its default database server and storage server settings.

Property	Default Provisioning
Shape Name	Exadata.X9M
Number of Compute Nodes	2
Total Maximum Number of Enabled CPU Cores	252
Total RAM Capacity (GB)	2,780
Persistent Memory (TB)	4.5
Maximum Local Storage per Database Server (GB)	2243
Number of Exadata Storage Servers	3
Maximum Database Size, No Local Backup (TB)	152

For more details, see the Autonomous AI Database on Dedicated Exadata Infrastructure X9M data sheet.

Autonomous AI Database on Exadata Cloud@Customer

For Autonomous AI Databases deployed on Exadata Cloud@Customer, you can select any of the following shapes while provisioning an X9M-2 system. The table below lists the characteristics of the Exadata X9M-2 systems available in Autonomous AI Database on Dedicated Exadata Infrastructure deployed on Exadata Cloud@Customer.

Property	Exadata X9M-2 Base Rack	Exadata X9M-2 Quarter Rack	Exadata X9M-2 Half Rack	Exadata X9M-2 Full Rack
Shape Name	ExadataCC.BaseX9M	ExadataCC.X9M	ExadataCC.X9M	ExadataCC.X9M
Number of Compute Nodes	2	2	4	8
Total Maximum Number of Enabled CPU Cores	48	124	248	496

Property	Exadata X9M-2 Base Rack	Exadata X9M-2 Quarter Rack	Exadata X9M-2 Half Rack	Exadata X9M-2 Full Rack
Total RAM Capacity (GB)	656	2,780	5,560	11,120
Total Persistent Memory Capacity (TB)	0	4.5	9	18
Maximum Local Storage per Database Server (GB)	1077	2243	2243	2243
Number of Exadata Storage Servers	3	3	6	12
Maximum Database Size, No Local Backup (TB)	59	152	305	610
Maximum Database Size, Local Backup (Exadata Cloud@Customer only) (TB)	29	76	152	305

For more details, see the Oracle Exadata Database Service on Exadata Cloud@Customer X9M data sheet.

Exadata X8M-2 Systems

Autonomous AI Database on Oracle Public Cloud

For Autonomous AI Databases deployed on Oracle Cloud, X8M-2 systems support elastic configurations instead of fixed shapes such as quarter rack, half rack, or full rack. X8M-2 systems are currently provisioned with a default configuration of 2 database servers and 3 storage servers that is equivalent to a quarter rack.

Should you need to provision an X8M-2 system with a different configuration, you have the flexibility to choose another value for database servers and storage servers within a valid range. The acceptable range of values is 2 to 32 for database servers and 3 to 64 for storage servers.

To provision an X8M-2 system successfully, the service limits for *Exadata X8M Database Server Count* and *Exadata X8M Storage Server Count* in your tenancy should be at least 2 and 3, respectively. Request a service limit increase from the Oracle Cloud Infrastructure console if required. Observe that the Exadata X8M Database Server Count and Exadata X8M Storage Server Count limits are defined as counts per *availability domain*. Refer to Request a Service Limit Increase for more details.

The table below lists the characteristics of an Exadata X8M-2 system on Oracle Cloud with its default database server and storage server settings.

Property	Default Provisioning
Shape Name	Exadata.X8M
Number of Compute Nodes	2
Total Maximum Number of Enabled CPU Cores	100

Property	Default Provisioning
Total RAM Capacity (GB)	2,780
Persistent Memory (TB)	4.5
Maximum Local Storage per Database Server (GB)	2243
Number of Exadata Storage Servers	3
Maximum Database Size, No Local Backup (TB)	119

For more details, see the Oracle Exadata Cloud Service X8M data sheet.

Autonomous AI Database on Exadata Cloud@Customer

For Autonomous AI Databases deployed on Exadata Cloud@Customer, you can select any of the following shapes while provisioning an X8M-2 system. The table below lists the characteristics of the Exadata X8M-2 systems available in Autonomous AI Database on Dedicated Exadata Infrastructure deployed on Exadata Cloud@Customer.

Property	X8M-2 Base Rack	X8M-2 Quarter Rack	X8M-2 Half Rack	X8M-2 Full Rack
Shape Name	ExadataCC.BaseX8M	ExadataCC.X8M	ExadataCC.X8M	ExadataCC.X8M
Number of Compute Nodes	2	2	4	8
Total Maximum Number of Enabled CPU Cores	48	100	200	400
Total RAM Capacity (GB)	656	2,780	5,560	11,120
Persistent Memory (TB)	0	4.5	9	18
Maximum Local Storage per Database Server (GB)	900	2340	2340	2340
Number of Exadata Storage Servers	3	3	6	12
Maximum Database Size, No Local Backup (TB)	59	119	239	479
Maximum Database Size, Local Backup (Exadata Cloud@Customer) (TB)	29	59	119	239

For more details, see the Oracle Exadata Database Service on Exadata Cloud@Customer X8M data sheet.

Exadata X8-2 Systems

The following table lists the characteristics of the Exadata X8-2 systems available in Autonomous AI Database on Dedicated Exadata Infrastructure.

Specification	Exadata X8-2 Base Rack	Exadata X8-2 Quarter Rack	Exadata X8-2 Half Rack	Exadata X8-2 Full Rack
Shape Name	Exadata.Base3.48	Exadata.Quarter3.100	Exadata.Half3.200	Exadata.Full3.400
Number of Compute Nodes	2	2	4	8
Total Maximum Number of Enabled CPU Cores	48	100	200	400
Total RAM Capacity (GB)	720	1,440 GB	2,880 GB	5,760 GB
Number of Exadata Storage Servers	3	3	6	12
Maximum Database Size, No Local Backup (TB)	59	119	238	476
Maximum Database Size, Local Backup (Exadata Cloud@Customer only) (TB)	29	59	119	239

For more details, see Oracle Exadata Database Machine X8-2 data sheet.

Exadata X7-2 Systems

The following table lists the characteristics of the Exadata X7-2 systems available in Autonomous AI Database on Dedicated Exadata Infrastructure.

Note

For Oracle Autonomous AI Database on Dedicated Exadata Infrastructure on Exadata Cloud@Customer, the Oracle Exadata X7-2 models listed below are supported only for the purposes of upgrading from Gen 1 (OCI-C) Exadata Cloud@Customer to Gen 2 Exadata Cloud@Customer.

Specification	Exadata X7-2 Base Rack	Exadata X7-2 Quarter Rack	Exadata X7-2 Half Rack	Exadata X7-2 Full Rack
Shape Name	Exadata.Base2.44	Exadata.Quarter2.92	Exadata.Half2.184	Exadata.Full2.368
Number of Compute Nodes	2	2	4	8
Total Maximum Number of Enabled CPU Cores	44	92	184	368
Total RAM Capacity (GB)	480	1,440	2,880	5,760
Number of Exadata Storage Servers	3	3	6	12
Maximum Database Size, No Local Backup (TB)	59	85.5	171.1	342.1

Specification	Exadata X7-2 Base Rack	Exadata X7-2 Quarter Rack	Exadata X7-2 Half Rack	Exadata X7-2 Full Rack
Maximum Database Size, Local Backup (Exadata Cloud@Customer only) (TB)	29	42.8	85.5	171.1

For more details, see Oracle Exadata Database Machine X7-2 data sheet.

CPU Overprovisioning with Autonomous AI Database on Dedicated Exadata Infrastructure

You can provision more Autonomous AI Databases than the available physical CPUs on Dedicated Exadata Infrastructure. This is known as CPU overprovisioning.

Note

CPU overprovisioning is supported for the OCPU compute type only.

About CPU Overprovisioning

CPU overprovisioning allows you to create more databases at the system level and run more databases on each infrastructure instance.

While provisioning Autonomous AI Databases, you can assign a fractional OCPU value (0.1 to 0.9 OCPUs with an increment of 0.1) to databases that do not need an entire OCPU. You can optimize the OCPU consumption by allocating a fraction of an OCPU to databases with minimal or non-performance critical workloads used for non-production databases such as development or test environments. Provisioning a test database with a less than 1 OCPU, for example, 0.1 OCPUs, lets you retain the remaining 0.9 OCPUs for other databases. Thus, you can provision more databases than the available CPUs. For example, an Exadata Infrastructure with 100 CPUs can cater to 1000 Autonomous AI Databases with CPU overprovisioning.

For databases using one or more OCPUs, you must increment the number of assigned CPUs by multiples of 1 or more OCPUs. For example, you cannot assign 3.5 OCPUs to a database. The next available number of OCPUs above 3 is 4. You can also assign as little as 32 GB of storage to each Autonomous AI Database. The lowered minimum requirements for OCPU and storage resources allow you to overprovision databases and run more databases on each infrastructure instance.

Implementing CPU Overprovisioning

Databases provisioned with OCPU overprovisioning can only connect to:

- The `tp` and `low` services for the Autonomous AI Transaction Processing workloads.
- The `low` services for Autonomous AI Lakehouse workloads.

The resources allocated to databases provisioned with OCPU overprovisioning are calculated proportionally based on the number of OCPUs chosen. For example, an Autonomous AI

Database with 0.3 OCPUs will get memory and concurrent statement allocations that is 30% of the allocation for a single OCPU Autonomous AI Database. See Predefined Database Service Names for Autonomous AI Databases for more details.

When creating a new Autonomous AI Database instance or cloning an existing Autonomous AI Database instance or its backup:

- CPU overprovisioning is not supported for databases using ECPUs. The minimum ECPU value for an Autonomous AI Database is 2.
- You can assign a fractional *OCPU* value from 0.1 to 0.9 (in increments of 0.1 OCPUs) to databases that do not need an entire OCPU. For databases that need one or more OCPUs, you must specify the number of assigned OCPUs as an integer. For example, you cannot assign 3.5 OCPUs to a database. The next available number of OCPUs above 3 is 4.
- While cloning from a backup, you can resize the CPU to a fractional value *only* after the clone, if needed. Refer to Clone Autonomous AI Database from Backup for instructions on cloning from a backup.

When scaling an Autonomous AI Database up or down:

- Scaling up from a fractional value to a full OCPU does not impact the predefined database services you can connect to. You can only connect to the `tp` and `low` services for the Autonomous AI Transaction Processing workloads and the `low` services for the Autonomous AI Lakehouse workloads. Refer to Add CPU or Storage Resources to an Autonomous AI Database for instructions on scaling up your database.
- Scaling down to a fractional value from a full OCPU does not impact the predefined database services you can connect to. Despite being on an overprovisioned OCPU, you can continue to connect to all the predefined database services, as done before scaling down. Refer to Remove CPU or Storage Resources from an Autonomous AI Database for instructions on scaling down your database.

Auto-scaling enables an Autonomous AI Database to use up to 3 times more CPU and IO resources than its allocated CPU count. When auto-scaling is applied on databases created on overprovisioned OCPUs, if 3 times the CPU count results in a fractional value, it will be rounded to the next whole number. See CPU Allocation When Auto-Scaling for more details.

Known Issues

Set `large_pool_size` to 20G to avoid connection problem while using the shared server configuration. (32840236)

Cannot connect to dedicated Autonomous Database from 12102/12201 jdbc thin driver clients (29251641)

Cannot import data from dedicated Autonomous Database to a local db instance (29276889)

CQN is not usable due to missing privs. (29498851)

ORA-600 - Unable to turn ON Application Continuity(AC) using `DBMS_APP_CONT.ENABLE_AC` (29544951)

ORA-12012 when executing APEX jobs (28799487)