# Oracle Cloud

# Oracle Autonomous Database on Dedicated Exadata Infrastructure

Release

January 2024

Oracle Cloud Oracle Autonomous Database on Dedicated Exadata Infrastructure, Release

Copyright © 2020, 2024, Oracle and/or its affiliates.

Primary Author: Anjani Surekha Pothula

Contributing Authors: Jonah Matishak

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.

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

# Contents

## 1 Overview


New Feature Announcements 1-1
- March 2024 1-1
- February 2024 1-2
- January 2024 1-4

About Autonomous Database on Dedicated Exadata Infrastructure 1-5
- Key Features of Autonomous Database on Dedicated Exadata Infrastructure 1-6
- Deployment Choices for Autonomous Database on Dedicated Exadata Infrastructure 1-7
- Components of Autonomous Database on Dedicated Exadata Infrastructure 1-8
- User Roles Associated with Autonomous Database on Dedicated Exadata Infrastructure 1-9

Autonomous Database for Developers 1-10
- Requirements 1-10
- Provisioning Workflow 1-10
- Specifications 1-11
- Excluded Features 1-11
- Supported Features 1-11

Compute Management and Billing in Autonomous Database on Dedicated Exadata Infrastructure 1-12
- Compute Models in Autonomous Database on Dedicated Exadata Infrastructure 1-12
- Compute Management in Autonomous Database 1-13
- CPU Allocation When Auto-Scaling 1-14
- CPU Billing Details 1-15
- How Compartment Quotas Affect CPU Management 1-16
- How VM Cluster Nodes Affect CPU Management 1-16

Get Help and Contact Oracle Support 1-17
- Create a Service Request in My Oracle Support 1-18
- Set Up Access to My Oracle Support 1-19


## 2 Quick Start


Before You Begin with Autonomous Database 2-1

Workflows 2-1

Set up Infrastructure Resources for Autonomous Database on Dedicated Exadata Infrastructure 2-1
Create and Use Autonomous Database 2-2
Develop Applications with Autonomous Database 2-3
Access Autonomous Database in the Oracle Cloud Infrastructure Console 2-3
LiveLabs for Autonomous Database on Dedicated Exadata Infrastructure 2-4

## 3 Features

High Availability 3-1
Protect Critical Databases from Failures and Disasters Using Autonomous Data Guard 3-1
About Autonomous Data Guard 3-1
Autonomous Data Guard Configuration Options 3-6
How Autonomous Data Guard Affects Standard Management Operations 3-8
Step-by-Step Guides 3-9
Backup and Restore Autonomous Database on Dedicated Exadata Infrastructure 3-9
About Backup and Recovery 3-9
Backup and Restore Operations 3-11
Backup and Restore Events 3-12
High Performance 3-13
High Performance Features in Autonomous Database on Dedicated Exadata Infrastructure 3-13
Predefined Database Services 3-14
Connection Pools 3-14
Special-Purpose Connection Features 3-14
SQL Performance Tuning Features 3-15
SQL Tracing 3-15
Optimizer Statistics 3-16
Optimizer Hints 3-16
Automatic Indexing 3-17
Fast Ingest 3-18
Predefined Job Classes with Oracle Scheduler 3-19
Performance Monitoring and Tuning Tools 3-19
Use SQL Tracing on Autonomous Database on Dedicated Exadata Infrastructure 3-21
Configure SQL Tracing on Autonomous Database 3-22
Enable SQL Tracing on Autonomous Database 3-23
Disable SQL Tracing 3-24
View Trace File Saved to Cloud Object Store on Autonomous Database 3-24
View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database 3-25
Governance 3-26
Tagging Autonomous Database Cloud Resources 3-26
Importance of Tagging 3-26

Adding Tags 3-27
Oracle Standard Tags 3-28
Oracle Application Name Tags 3-30
Manage Resource Availability with Compartment Quotas 3-31
Quotas for Autonomous Database on Oracle Public Cloud 3-32
Quotas for Autonomous Database on Exadata Cloud@Customer 3-33
Continuous Availability 3-34
Code for Continuous Availability 3-34
Application Configuration Checklist 3-35
Connect Using Database Services 3-35
Configure Connection String for High Availability 3-35
Use Fast Application Notification (FAN) 3-36
Use Recommended Practices to Allow Draining 3-37
Enable Transparent Application Continuity (TAC) or Application Continuity (AC) 3-38
Configure Drivers for Continuous Availability 3-40
Configure JDBC Thin Driver 3-40
Configure OCI (Oracle Call Interface) Driver 3-41
Configure ODP.NET Unmanaged Provider Driver 3-42
Enable Service Attributes for Failover 3-42
Configure Service Attributes for Maintenance 3-43
Configure Clients for FAN Including Optional Wallets 3-43
Change Management 3-45
Service Maintenance of Autonomous Database on Dedicated Exadata Infrastructure 3-45
Service Maintenance Types 3-46
Specify When Maintenance Can Occur 3-47
Specify What Kind of Patches Apply 3-52
View and Manage Already Scheduled Maintenance 3-53
Automatic Queuing of Maintenance Events 3-54
View Past Maintenance Events 3-55
Monitor Service Maintenance Events 3-55
Service Monitoring 3-56
Plan and Monitor Capacity 3-56
Resource Terminology 3-56
Resource Limits 3-57
Resource Usage Tracking 3-58
Resource Usage Visualizations 3-59
Exadata System Shapes 3-62
Monitor Databases with Autonomous Database Metrics 3-63
Prerequisites 3-63
Using the Console 3-64
Using the API 3-64

Available Metrics: oci_autonomous_database 3-64
Monitor Database Performance with Performance Hub 3-69
Use Database Actions to Manage and Monitor Databases 3-72
Use Database Management Service to Monitor Databases 3-72
Use Enterprise Manager to Manage and Monitor Databases 3-73
Use Operations Insights on Autonomous Database 3-74
About Operations Insights 3-74
Use Work Requests to Monitor Long-Running Operations 3-75
Events and Notifications to Watch Resource Activity 3-75
Security 3-76
Security Features in Autonomous Database on Dedicated Exadata Infrastructure 3-76
Configuration Management 3-77
Data Encryption 3-77
Auditing 3-78
Access Control 3-78
Certificate Management 3-78
Data Protection 3-81
Sensitive Data Discovery and Data Masking 3-82
Regulatory Compliance Certification 3-83
Configuration Management in Autonomous Database on Dedicated Exadata Infrastructure 3-84
Autonomous Virtual Machine Hardening 3-84
Configuration Drift Management 3-85
File Integrity and Intrusion Monitoring 3-85
Autonomous VM Vulnerability Scanning and Response 3-85
Endpoint, Malware, and Ransomware Protection 3-86
Security Incident Management 3-86
Data Encryption in Autonomous Database on Dedicated Exadata Infrastructure 3-87
Encryption of Data at Rest 3-87
Encryption of Data in Transit 3-87
Auditing Capabilities in Autonomous Database 3-88
Auditing Service Level Activities 3-88
Auditing Database Activities 3-88
Auditing Autonomous VM Activities 3-89
Auditing Oracle Operator Activities 3-90
Access Control 3-92
Oracle Cloud User Access Control 3-92
Client Access Control 3-94
Database User Access Control 3-95
Privileged Access Management (PAM) 3-98
IAM Policies for Autonomous Database on Dedicated Exadata Infrastructure 3-99

Policy Details for Autonomous Database 3-99
Policies to Manage Exadata Infrastructure Resources 3-110
Policies to Manage Autonomous Exadata VM Clusters 3-110
Policies to Manage Autonomous Container Databases 3-111
Policies to Manage Autonomous Databases 3-112
Master Encryption Keys in Autonomous Database on Dedicated Exadata Infrastructure 3-113
Prepare to Use Customer-Managed Keys in the Vault Service 3-113
Prepare to Use Customer-Managed Keys in Oracle Key Vault 3-116
Use Customer-Managed Keys 3-116
Features for Experienced Database Users 3-117
Oracle Database Features in Autonomous Database on Dedicated Exadata Infrastructure 3-117
Latest Autonomous Database Features 3-118
The ADMIN User and the SYS User 3-119
Database Initialization Parameters 3-120
SYSDATE_AT_DBTIMEZONE 3-122
Oracle Database Features with Limited Support in Autonomous Database on Dedicated Exadata Infrastructure 3-124
SQL Commands 3-125
Data Types 3-128
Database In-Memory 3-128
Manually Created External Tables 3-129
Oracle LogMiner 3-130
Limitations on the Use of Oracle Spatial 3-130
Oracle Text 3-131
Oracle XML DB 3-131
Oracle Flashback and Restore Points 3-132
Database Replay 3-133
Database PL/SQL Packages 3-133
Oracle Database Features That Are Not Supported in Autonomous Database on Dedicated Exadata Infrastructure 3-134
Character Set Selection for Autonomous Database 3-135
Use Oracle Graph with Autonomous Database on Dedicated Exadata Infrastructure 3-136
Build Applications Using Graph APIs 3-136
Use Oracle Spatial in Autonomous Database 3-137
About Oracle Spatial 3-137
Limitations on the Use of Oracle Spatial 3-137
Use Advanced Queuing with Autonomous Database 3-138
Use JSON Documents with Autonomous Database on Dedicated Exadata Infrastructure 3-139

# 4 Tasks


About Connecting to Autonomous Database on Dedicated Exadata Infrastructure 4-1
- Oracle Call Interface (OCI) Connections and Wallets 4-2
- Predefined Database Service Names for Autonomous Databases 4-2
- Connect to Autonomous Database Using a Client Application 4-5
- Connect to Autonomous Database Using Oracle Database Tools 4-6
- Connect with Microsoft .NET, Visual Studio, and Visual Studio Code 4-7
- Connect to Autonomous Database with Scripting Languages 4-7
- Connect with Oracle Cloud Infrastructure FastConnect 4-8

Migrate 4-8
- Attach External Storage to Autonomous Database 4-8
  - Attach Network File Storage to Autonomous Database on Dedicated Exadata Infrastructure 4-8
  - Attach External File Storage to Autonomous Database on Dedicated Exadata Infrastructure 4-12
- Move Data Into Autonomous Database 4-17
  - Migrate Data into Autonomous Database on Dedicated Exadata Infrastructure 4-17
  - Loading Data into Autonomous Database on Dedicated Exadata Infrastructure 4-21
  - Load Data from Files in the Cloud 4-25
  - Use Oracle Data Pump 4-30
  - Use SQL*Loader 4-34
  - Migrate to Autonomous Database Using Oracle Database Migration Service 4-35
  - Use Oracle GoldenGate to Replicate Data with Autonomous Database 4-36
- Export Data from Autonomous Database 4-37
  - Use Oracle Data Pump to Export Data from Autonomous Database 4-37
  - Export Data to a Directory 4-44
  - Export Data to Object Store as Text Using DBMS_CLOUD.EXPORT_DATA 4-49

Manage Users 4-57
- Managing Database Users 4-57
  - Create Database Users 4-57
  - Unlock or Change the ADMIN Database User Password 4-58
  - Manage Database User Privileges 4-59
  - Remove Database Users 4-60
- Manage User Profiles with Autonomous Database 4-60
  - Manage Password Complexity on Autonomous Database 4-61
  - Gradual Database Password Rollover for Applications 4-63
- Use Oracle Database Vault to Manage Database User Privileges 4-63
  - Configure and Enable Oracle Database Vault 4-64
- Use Identity and Access Management (IAM) Authentication with Autonomous Database 4-64

About Identity and Access Management (IAM) Authentication with Autonomous Database 4-65
Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database 4-66
Change External Identity Providers on Autonomous Database 4-67
Create Identity and Access Management (IAM) Groups and Policies for IAM Users 4-68
Add IAM Users on Autonomous Database 4-70
Add IAM Roles on Autonomous Database 4-72
Create IAM Database Password for IAM Users 4-73
Connect to Autonomous Database with Identity and Access Management (IAM) Authentication 4-73
Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password 4-75
Configuring a Client Connection for SQL*Plus That Uses an IAM Token 4-76
Use Instance Principal to Access Autonomous Database with Identity and Access Management (IAM) Authentication 4-78
Configure Proxy Authentication 4-78
Use Database Link with IAM Authenticated Users 4-80
Disable Identity and Access Management (IAM) Authentication on Autonomous Database 4-80
Notes for Using Autonomous Database Tools with Identity and Access Management (IAM) Authentication 4-80
About Kerberos Authentication 4-81
Components of the Kerberos Authentication System 4-81
Notes About Kerberos Authentication on Autonomous Database 4-82
Enable Kerberos Authentication on Autonomous Database 4-82
Disable Kerberos Authentication on Autonomous Database 4-84
Authenticate and Authorize Microsoft Azure AD Users for Autonomous Database 4-84
About Authorizing Microsoft Azure AD Users for an Oracle Autonomous Database on Dedicated Exadata Infrastructure 4-85
General Process of Integrating Microsoft Azure AD with Oracle Autonomous Database on Dedicated Exadata Infrastructure 4-86
Configure the Oracle Autonomous Database on Dedicated Exadata Infrastructure for Microsoft Azure AD Integration 4-87
Map Oracle Database Schemas and Roles 4-90
Configure Client Connections to Azure ADs 4-91
Trace Files Used for Troubleshooting Connections 4-92
Use Microsoft Active Directory with Autonomous Database on Dedicated Exadata Infrastructure 4-93
Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous Database 4-93
Configure CMU with Microsoft Active Directory on Autonomous Database 4-94
Configure CMU with Microsoft Active Directory on Exadata Cloud@Customer 4-97
Add Microsoft Active Directory Roles on Autonomous Database 4-100

Add Microsoft Active Directory Users on Autonomous Database 4-101
Connect to Autonomous Database with Active Directory User Credentials 4-103
Verify Active Directory User Connection Information with Autonomous Database 4-103
Remove Active Directory Users and Roles on Autonomous Database 4-104
Disable Active Directory Access on Autonomous Database 4-105
Limitations with Microsoft Active Directory on Autonomous Database 4-105
Develop 4-105
Developing Applications with Autonomous Database 4-105
Development Workflow 4-106
Create a Database 4-106
Create a Directory 4-107
Manage Directories 4-108
Configure a Development System to Access the Database 4-109
Define or Load the Database Structure 4-111
Download, Install and Configure Oracle Database JDBC Driver 4-111
Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections 4-112
The Basics of Building an Application 4-113
Build Database Applications with Autonomous Database 4-114
Build a Java Application 4-114
Build a .NET Application with Visual Studio and Visual Studio Code 4-117
Build a Node.js Application 4-119
Build a PHP Application 4-122
Build a Python Application 4-125
Build a Ruby Application 4-128
Build an APEX (Oracle Application Express) Application 4-131
Developing RESTful Services in Autonomous Database 4-134
Send Email and Notifications from Autonomous Database 4-135
Send Email on Autonomous Database on Dedicated Exadata Infrastructure 4-135
Send Microsoft Teams Notifications from Autonomous Database on Dedicated Exadata Infrastructure 4-142
Send Slack Notifications from Autonomous Database on Dedicated Exadata Infrastructure 4-146
Manage Data 4-149
About Cloning Autonomous Database on Dedicated Exadata Infrastructure 4-149
Clone Types 4-149
Clone Sources 4-149
Clone Requirements 4-150
Clone Limitations 4-151
Step-by-Step Guides 4-152
Integrate 4-153
Analyzing and Visualizing Data 4-153
Use Oracle Analytics Desktop to Analyze and Visualize Data 4-153

Use Oracle Analytics Cloud to Analyze and Visualize Data 4-153
Querying External Data with Autonomous Database on Dedicated Exadata Infrastructure 4-154
Query External Data 4-155
Validate External Data 4-156
View Logs for Data Validation 4-157
Oracle Extensions for IDEs 4-158
Use Oracle Cloud Infrastructure Toolkit for Eclipse 4-158
Use Oracle Developer Tools for Visual Studio 4-159
Use Oracle Developer Tools for VS Code 4-160
Connect Power BI and Microsoft Data Tools to Autonomous Database on Dedicated Exadata Infrastructure 4-160
Use Select AI to Generate SQL from Natural Language Prompts 4-162
Usage Guidelines 4-162
About SQL Generation 4-163
Use DBMS_CLOUD_AI to Configure AI Profiles 4-163
Requirements to Configure DBMS_CLOUD_AI Package 4-163
Configure DBMS_CLOUD_AI Package 4-164
Create and Set an AI Profile 4-166
Use AI Keyword to Enter Prompts 4-167
Terminology 4-168

5 Use Cases

Configuration Example: Network Isolation by Subnet 5-1
Step 1. Create Compartments 5-2
Step 2. Create the VCN and Subnets 5-3
Step 3. Create Groups 5-5
Step 4. Create Policies 5-6
Step 5. Assign Fleet Administrators 5-7
Step 6. Create the Exadata Infrastructure Resource 5-7
Step 7. Create the Autonomous Exadata VM Cluster Resource 5-7
Step 8. Create Autonomous Container Database Resources 5-7
Step 9. Assign Database Administrators 5-8
Notifications Example: Emails for Maintenance Events 5-8
Step 1. Create a Notifications Topic 5-9
Step 2. Add a Subscription to the Notifications Topic 5-9
Step 3. Add an Events Service Rule 5-10

6 How-to Guides

Create and Manage Exadata Infrastructure Resources 6-1

Create an Exadata Infrastructure Resource 6-1
View Details of an Exadata Infrastructure Resource 6-4
Move an Exadata Infrastructure Resource to a Different Compartment 6-6
Terminate an Exadata Infrastructure Resource 6-7
View a List of Exadata Infrastructure Resources 6-7
Change the Maintenance Schedule of an Exadata Infrastructure Resource 6-8
View and Manage Scheduled Maintenance of an Exadata Infrastructure Resource 6-9
View Past Maintenance of an Exadata Infrastructure Resource 6-9
Create and Manage Autonomous Exadata VM Clusters 6-10
Create Autonomous Exadata VM Cluster 6-10
Prerequisites 6-10
Procedure 6-13
Manage Autonomous Exadata VM Clusters 6-18
View a List of Autonomous Exadata VM Clusters 6-18
View Details of an Autonomous Exadata VM Cluster 6-20
Change the License Type on an Autonomous VM Cluster 6-23
Move an Autonomous Exadata VM Cluster to Another Compartment 6-24
Manage Security Certificates for an Autonomous Exadata VM Cluster Resource 6-24
View Resource Usage for an Autonomous Exadata VM Cluster 6-26
Manage Autonomous Exadata VM Cluster Resources 6-28
Terminate an Autonomous Exadata VM Cluster 6-29
Manage Exadata Infrastructure Maintenance 6-29
Change the Maintenance Schedule of an Autonomous Exadata VM Cluster 6-29
View and Manage Scheduled Maintenance of an Autonomous Exadata VM Cluster 6-30
View Past Maintenance of an Autonomous Exadata VM Cluster 6-31
Create and Manage Autonomous Container Databases 6-31
About Autonomous Container Database 6-31
Autonomous Container Database Requirements 6-32
Database Features Managed from Autonomous Container Database 6-32
Autonomous Container Database Management Operations 6-38
Create an Autonomous Container Database 6-39
View Details of an Autonomous Container Database 6-47
View Resource Usage for an Autonomous Container Database 6-49
Restart an Autonomous Container Database 6-50
Move an Autonomous Container Database to a Different Compartment 6-51
View a List of Autonomous Container Databases 6-51
View a List of Autonomous Container Databases in a Compartment 6-51
View a List of Autonomous Container Databases in an Autonomous Exadata VM Cluster Resource 6-53
Configure and Manage Autonomous Data Guard 6-55
Enable Autonomous Data Guard on an Autonomous Container Database 6-55

View the Status of an Autonomous Data Guard Configuration 6-58
Switch Roles in an Autonomous Data Guard Configuration 6-59
Fail Over to the Standby in an Autonomous Data Guard Configuration 6-59
Reinstate the Disabled Standby in an Autonomous Data Guard Configuration 6-60
Update Autonomous Data Guard Settings 6-61
Convert Physical Standby to Snapshot Standby 6-62
Convert Snapshot Standby to Physical Standby 6-63
Update Autonomous Container Database Maintenance Preferences 6-63
View and Manage Scheduled Maintenance of an Autonomous Container Database 6-65
View Past Maintenance of an Autonomous Container Database 6-65
Schedule a Quarterly Maintenance Update 6-66
Edit Autonomous Container Database Backup Settings 6-67
Terminate an Autonomous Container Database 6-70
Create and Manage Autonomous Databases 6-71
Create an Autonomous Database 6-71
View Details of an Autonomous Database 6-76
Start, Stop, and Restart an Autonomous Database 6-77
Stop an Autonomous Database 6-77
Start an Autonomous Database 6-77
Restart an Autonomous Database 6-78
Enable or Disable Database In-Memory 6-78
Enable or Disable Auto Scaling of an Autonomous Database 6-79
Manage CPU or Storage Resources of an Autonomous Database 6-80
Add CPU or Storage Resources to an Autonomous Database 6-80
Remove CPU or Storage Resources from an Autonomous Database 6-82
Move an Autonomous Database to a Different Compartment 6-83
Clone an Autonomous Database on Dedicated Exadata Infrastructure 6-83
Clone an Autonomous Database Instance 6-84
Clone Autonomous Database from Backup 6-88
Managing Primary and Standby Databases in an Autonomous Data Guard Configuration 6-94
Backup an Autonomous Database on Dedicated Exadata Infrastructure Manually 6-95
Create a Long-Term Backup 6-96
Procedure 6-96
Manage Long-Term Backups 6-98
Restore and Recover Autonomous Database on Dedicated Exadata Infrastructure 6-99
View Autonomous Database Metrics for a Database 6-100
Procedure 6-100
View Autonomous Database Metrics for a Database 6-101
Procedure 6-101
Enable Operations Insights for an Autonomous Database 6-101

Terminate an Autonomous Database on Dedicated Exadata Infrastructure 6-103
Connect to Autonomous Database 6-103
Download Client Credentials 6-103
View Connection Strings for an Autonomous Database 6-104
Prepare for Oracle Call Interface (OCI), ODBC, and JDBC OCI Connections 6-105
Prepare for JDBC Thin Connections 6-106
Connect with Oracle Database Actions 6-107
Connect to Autonomous Database on Dedicated Exadata Infrastructure with Oracle SQL Developer 6-108
Connect with Oracle SQL Developer Version 18.2 or Later 6-108
Connect with Oracle SQL Developer Versions Earlier Than 18.2 6-110
Connect with Oracle SQLcl 6-111
Connect with SQL*Plus 6-113
Connect Python Applications Using mTLS 6-114
Install Python and the python-oracledb Driver 6-115
Obtain Security Credentials (Oracle Wallet) and Enable Network Connectivity 6-117
Run Python Application with python-oracledb Thin Mode (mTLS) 6-117
Run Python Application with python-oracledb Thick Mode (mTLS) 6-120
Connect with Node.js and other Scripting Languages 6-122
Secure Autonomous Database 6-125
Rotate Encryption Keys 6-125
Rotate the Encryption Key of an Autonomous Container Database 6-125
Rotate the Encryption Key of an Autonomous Database 6-126
Register or Deregister a Database with Data Safe 6-127
Set Access Control List for an Autonomous Database 6-127

## 7 Reference

DBMS_CLOUD Package 7-1
DBMS_CLOUD for Access Management 7-1
Prerequisites 7-1
DBMS_CLOUD Subprograms for Access Management 7-2
CREATE_CREDENTIAL Procedure 7-3
DROP_CREDENTIAL Procedure 7-5
UPDATE_CREDENTIAL Procedure 7-5
DBMS_CLOUD for Objects and Files 7-7
Prerequisites 7-7
DBMS_CLOUD Subprograms for Objects and Files 7-8
COPY_COLLECTION Procedure 7-10
COPY_DATA Procedure 7-12
COPY_DATA Procedure for Avro, ORC, or Parquet Files 7-14
COPY_OBJECT Procedure 7-16

CREATE_EXTERNAL_TABLE Procedure 7-17
CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files 7-19
CREATE_HYBRID_PART_TABLE Procedure 7-22
DELETE_ALL_OPERATIONS Procedure 7-24
DELETE_FILE Procedure 7-24
DELETE_OBJECT Procedure 7-25
EXPORT_DATA Procedure 7-26
GET_OBJECT Procedure and Function 7-29
LIST_FILES Function 7-31
LIST_OBJECTS Function 7-32
MOVE_OBJECT Procedure 7-33
PUT_OBJECT Procedure 7-34
SYNC_EXTERNAL_PART_TABLE Procedure 7-36
VALIDATE_EXTERNAL_PART_TABLE Procedure 7-36
VALIDATE_EXTERNAL_TABLE Procedure 7-37
VALIDATE_HYBRID_PART_TABLE Procedure 7-38
DBMS_CLOUD REST APIs 7-40
Prerequisites 7-40
DBMS_CLOUD REST APIs 7-41
DBMS_CLOUD REST API Overview 7-42
DBMS_CLOUD REST API Constants 7-42
DBMS_CLOUD REST API Results Cache 7-42
DBMS_CLOUD REST API Results cache_scope Parameter 7-43
DBMS_CLOUD REST API SESSION_CLOUD_API_RESULTS View 7-44
GET_RESPONSE_HEADERS Function 7-44
GET_RESPONSE_TEXT Function 7-45
GET_API_RESULT_CACHE_SIZE Function 7-45
SEND_REQUEST Function and Procedure 7-45
SET_API_RESULT_CACHE_SIZE Procedure 7-48
Format Parameter 7-48
DBMS_CLOUD Package Format Options for EXPORT_DATA 7-54
DBMS_CLOUD Avro, ORC, and Parquet Support 7-59
DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet 7-59
DBMS_CLOUD Package Avro to Oracle Data Type Mapping 7-60
DBMS_CLOUD Package ORC to Oracle Data Type Mapping 7-61
DBMS_CLOUD Package Parquet to Oracle Data Type Mapping 7-62
DBMS_CLOUD Package Oracle Data Type to Parquet Mapping 7-62
DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types 7-64
DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping 7-68
DBMS_CLOUD Exceptions 7-70

Cloud Object Storage URI Formats 7-71
Oracle Cloud Infrastructure Object Storage Native URI Format 7-71
Oracle Cloud Infrastructure Object Storage Swift URI Format 7-72
Oracle Cloud Infrastructure Object Storage URI Format Using Pre-Authenticated Request URL 7-73
Amazon S3 URI Format 7-74
Azure Blob Storage URI Format 7-74
Google Cloud Storage S3 Compatible URI Format 7-75
SODA Collection Metadata on Autonomous Database 7-75
SODA Default Collection Metadata on Autonomous Database 7-75
SODA Customized Collection Metadata on Autonomous Database 7-76
Autonomous Database Views 7-78
DBMS_CLOUD_AI Views 7-78
DBA_CLOUD_AI_PROFILES View 7-78
USER_CLOUD_AI_PROFILES View 7-78
DBA_CLOUD_AI_PROFILE_ATTRIBUTES View 7-79
USER_CLOUD_AI_PROFILE_ATTRIBUTES View 7-79
Summary of DBMS_CLOUD_ADMIN Subprograms 7-79
ATTACH_FILE_SYSTEM Procedure 7-80
DETACH_FILE_SYSTEM Procedure 7-81
DISABLE_EXTERNAL_AUTHENTICATION Procedure 7-82
ENABLE_EXTERNAL_AUTHENTICATION Procedure 7-83
DBMS_CLOUD_NOTIFICATION Package 7-87
Summary of DBMS_CLOUD_NOTIFICATION Subprograms 7-88
SEND_DATA Procedure 7-88
SEND_MESSAGE Procedure 7-92
DBMS_CLOUD_AI Package 7-95
Summary of DBMS_CLOUD_AI Subprograms 7-96
CREATE_PROFILE Procedure 7-97
DROP_PROFILE Procedure 7-98
ENABLE_PROFILE Procedure 7-99
DISABLE_PROFILE Procedure 7-99
SET_ATTRIBUTE Procedure 7-100
SET_PROFILE Procedure 7-100
GENERATE Function 7-101
Profile Attributes 7-105
Troubleshooting Autonomous Database on Dedicated Exadata Infrastructure 7-107
Unable to Access Master Encryption Keys 7-107
Unable to Access Vault 7-108
Backup to Network File System (NFS) is Failing 7-109
Unable to Mount Network File System (NFS) 7-109

Unable to write a file in Network File System (NFS) 7-110
Unable to get outbound traffic from APEX 7-110
Backup Retention Issues with ZDLRA 7-111
API Reference for Autonomous Database on Dedicated Exadata Infrastructure 7-111
API to Manage Exadata Infrastructure Resources 7-111
API to Manage Autonomous Exadata VM Clusters 7-113
API to Manage Autonomous Container Databases 7-114
API to Manage Autonomous Databases 7-115
API to Manage Autonomous Data Guard Configuration 7-117
API to Manage Autonomous Virtual Machines 7-118
Events for Autonomous Database on Dedicated Exadata Infrastructure 7-118
Cloud Exadata Infrastructure Event Types 7-119
Cloud Autonomous Exadata VM Cluster Event Types 7-124
Autonomous Container Database Event Types 7-128
Autonomous Database Event Types 7-131
Autonomous Data Guard Event Types 7-136
Availability Service Level Agreements (SLAs) 7-138
Certify Applications for Compatibility Between Autonomous Database Serverless and Dedicated 7-140
Feature Parity Between Oracle Public Cloud and Exadata Cloud@Customer 7-143
Characteristics of Infrastructure Shapes 7-144
Exadata Base Systems 7-144
Exadata X10M Systems 7-145
Exadata X9M-2 Systems 7-146
Exadata X8M-2 Systems 7-147
Exadata X8-2 Systems 7-148
Exadata X7-2 Systems 7-149
CPU Overprovisioning with Autonomous Database on Dedicated Exadata Infrastructure 7-150
About CPU Overprovisioning 7-150
Implementing CPU Overprovisioning 7-150
Known Issues 7-151
Previous Feature Announcements 7-152
New Feature Announcements (2023) 7-152
December 2023 7-152
November 2023 7-152
October 2023 7-153
September 2023 7-155
August 2023 7-156
July 2023 7-157
June 2023 7-158
May 2023 7-160

April 2023 7-163
March 2023 7-164
February 2023 7-165
New Feature Announcements 7-167
December 2022 7-167
November 2022 7-169
October 2022 7-169
August 2022 7-170
July 2022 7-171
June 2022 7-171
May 2022 7-172
April 2022 7-174
March 2022 7-175
February 2022 7-176
January 2022 7-177
New Feature Announcements (2021) 7-178
December 2021 7-178
November 2021 7-179
October 2021 7-179
September 2021 7-180
July 2021 7-180
June 2021 7-181
May 2021 7-181
April 2021 7-182
February 2021 7-183
January 2021 7-183
New Feature Announcements (2020) 7-183
December 2020 7-184
November 2020 7-184
October 2020 7-185
September 2020 7-185
August 2020 7-185
July 2020 7-186
June 2020 7-186
May 2020 7-187
April 2020 7-187
March 2020 7-187
February 2020 7-188
January 2020 7-189
New Feature Announcements (2019) 7-189
September 2019 7-189

August 2019 7-190
July 2019 7-190
June 2019 7-190

# 1 Overview

## New Feature Announcements

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

### 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.<br>See About Autonomous Container Database. |
| Free developer database instances | Customers who have subscribed to Oracle Exadata Database Service or Autonomous Database on either Dedicated Exadata Infrastructure or Exadata Cloud@Customer can create and use Autonomous Database for Developers instances.<br>Autonomous Database for Developers instances are free Autonomous Databases that developers can use to build and test new applications.<br>With Autonomous Database for Developers instances, you can try new Autonomous 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 Database.<br>Refer to Autonomous Database for Developers for more details. |

| Feature | Description |
|---|---|
| Shared Server configuration for Autonomous Database applications | Autonomous 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.<br>For more details, see the Shared Server Configuration section in Special-Purpose Connection Features. |
| Autonomous Database resource management | Autonomous 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):<br>• Database split threshold (CPU)<br>• Node failover reservation (%)<br>• Distribution affinity<br>This capability improves your resource management ability to either consolidate more databases or have the highest database availability.<br>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 Database on Dedicated Exadata Infrastructure to a directory or cloud Object Store. The export options are CSV, JSON, or XML.<br>See the following pages for more details:<br>• Export Data to a Directory Using DBMS_CLOUD.EXPORT_DATA<br>• Export Data to Object Store as Text Using DBMS_CLOUD.EXPORT_DATA<br>• EXPORT_DATA Procedure Reference<br>• 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.<br>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 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.<br>See Use Select AI to Generate SQL from Natural Language Prompts for more information.<br>To get a list of AI profiles and their attributes created in your Autonomous Database or schema, you can use the `DBMS_CLOUD_AI` views.<br>See DBMS_CLOUD_AI Views for more information. |
| Send Microsoft Teams notifications from Autonomous 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 Database to a Microsoft Teams Channel.<br>See Send Microsoft Teams Notifications from Autonomous Database for more information. |
| Send Slack notifications from Autonomous 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 Database. This feature enables you to send messages to a Slack channel or share the results of a query to a Slack channel.<br>See Send Slack Notifications from Autonomous Database for more information. |

| Feature | Description |
| --- | --- |
| 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.<br>*Within two weeks from the certificate rotation*, you can connect to your databases using the Autonomous Database client wallet you downloaded before or after the certificate rotation.<br>*After two weeks from the certificate rotation:*<br>• The database wallet *downloaded before the certificate rotation* is invalidated and can not be used to connect to your databases.<br>• The database wallet *downloaded within two weeks* from the certificate rotation remains active and can be used to connect to your databases.<br>• Any new database wallet *downloaded after two weeks* from the certificate rotation can be used to connect to your databases.<br>Refer to Certificate Management for complete details. |
| 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.<br>To subscribe to event notifications, see Use Events and Notifications to Watch Resource Activity.<br>Refer to Events for Autonomous Database on Dedicated Exadata Infrastructure for a complete list of events emitted by the Autonomous Database resources. |

# January 2024

| Feature | Description |
| --- | --- |
| Send email using `DBMS_CLOUD_NOTIFICATION` from Autonomous 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.<br>See Send Email on Autonomous Database on Dedicated Exadata Infrastructure for more information. |

| Feature | Description |
|---|---|
| Send Microsoft Teams notifications from Autonomous Databases deployed on Oracle Public Cloud. | On Oracle Public Cloud, you have the capability to send messages, alerts, or query results from an Autonomous Database to a Microsoft Teams Channel.<br>See Send Microsoft Teams Notifications from Autonomous Database for more information. |
| Send Slack notifications from Autonomous Databases deployed on Oracle Public Cloud. | On Oracle Public Cloud deployments, you have the capability to send Slack notifications from an Autonomous Database. This feature enables you to send messages or share the results of a query to a Slack channel.<br>See Send Slack Notifications from Autonomous Database for more information. |

# About Autonomous Database on Dedicated Exadata Infrastructure

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

Oracle Autonomous 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 Data Warehouse
- Autonomous Transaction Processing

### Autonomous Data Warehouse

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

### Autonomous Transaction Processing

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

## Key Features of Autonomous Database on Dedicated Exadata Infrastructure

Autonomous 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 Database technology. With Autonomous Database, you do not need to configure or manage any hardware or install any software. Autonomous 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 Database is designed as a "Load and Go" service to define tables, load data, and start running queries. Autonomous 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 Database is a fully elastic service. When you get started with Autonomous 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 Database to the exact compute and storage required or even shut off idle compute to save money.
- **Highly Available:** Autonomous 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 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 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 Database is built upon Oracle Database so that the applications and tools that support Oracle Database also support Autonomous Database. These tools and applications can either be in your data center or a public cloud and connect to Autonomous Database using standard SQL*Net connections.
- **Integration with Third Party Tools and Utilities:** With Autonomous 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 Database to easily migrate your data from MySQL, Postgre SQL, SQL Server, and other databases. Autonomous Database supports connectivity via SQL*Net, JDBC, and ODBC.
- **Enriched with Built-in Tools:** Autonomous 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 Database on Dedicated Exadata Infrastructure

Autonomous Database on Dedicated Exadata Infrastructure offers two 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 Databases on Oracle Public Cloud get dedicated system resources such as processor, memory, network, or storage to offer greater control over operational policies and customizations.
- **Exadata Cloud@Customer Infrastructure:** An Autonomous 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.

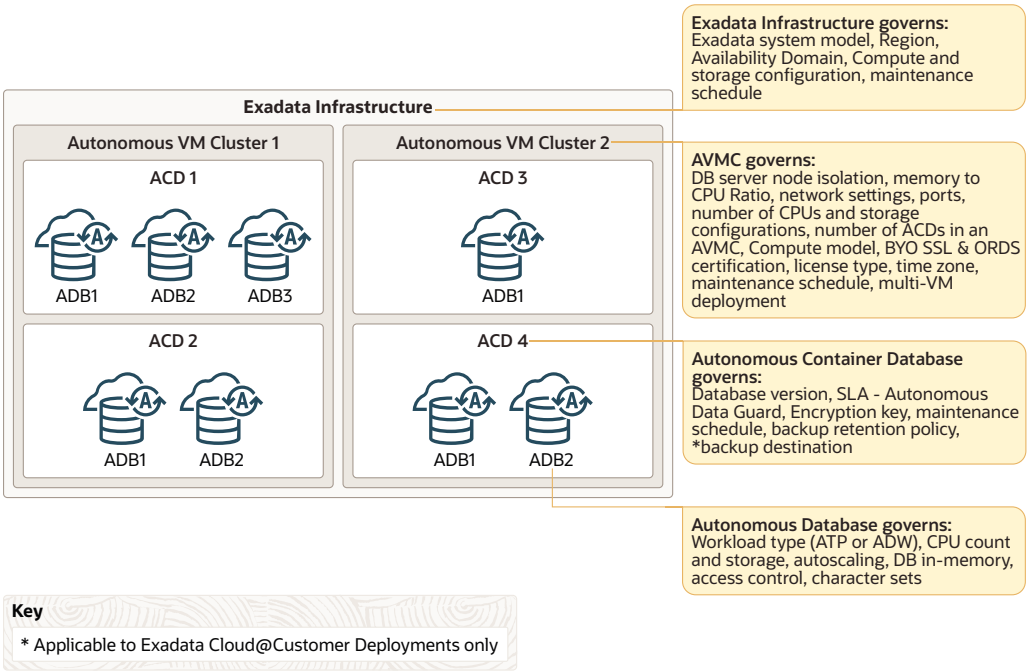
Both platforms support a standard set of features for Autonomous Databases with a few differences. All through this documentation, you will encounter the following badges as applicable:

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

For a glance at the features supported exclusively by Oracle cloud or Exadata Cloud@Customer deployment choices, refer to Feature Parity Between Oracle Cloud and Exadata Cloud at Customer.

# Components of Autonomous Database on Dedicated Exadata Infrastructure

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




Autonomous 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.
- An **Autonomous Exadata VM Cluster** 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 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 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.

  - Create both Autonomous Exadata VM Clusters and Exadata Database VM Clusters on a common Exadata Infrastructure resource.

> **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 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** resource provides a container for your Autonomous Databases. You can create multiple Autonomous Container Database resources in a single Autonomous Exadata VM Cluster resource, but you must create at least one before you can create any Autonomous Databases. For an overview of Autonomous Container Databases, their features, and how to use them, see About Autonomous Container Database.
- An **Autonomous Database** resource is a user database. You can create many Autonomous Databases in a single Autonomous Container Database resource.
- An **Autonomous Backup** resource is a backup of an Autonomous Database. Oracle creates these resources for you automatically. See Backup and Restore Autonomous Database for more details.

## User Roles Associated with Autonomous Database on Dedicated Exadata Infrastructure

The tasks involved in setting up and using the Autonomous 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 Databases. Additionally, they create and manage Oracle 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 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 Database. When creating an Autonomous 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 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 Database for Developers

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

With Autonomous Database for Developers instances, you can try new Autonomous 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 Database.

## Requirements

To create an Autonomous Database for Developers, you must have access to Oracle Exadata Database Service or Autonomous 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 Database on Dedicated Exadata Infrastructure
- Oracle Exadata Database Service on Dedicated Infrastructure
- Autonomous 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.

## Provisioning Workflow

You can provision an Autonomous 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 Database for Developers using them. Provisioning a developer database using the OCI console follows the same workflow as creating a regular Autonomous Database, as described in Create an Autonomous Database. Once created, the Autonomous Database for Developers instances appear with a **Developer label** in the list of Autonomous Databases on the OCI console.

## Specifications

Each developer database comes with the following specifications:

- **Compute:** Fixed 4 ECPUs, with no CPU scaling
- **Storage:** Fixed 32 GB ( ~ 20 GB of DATA)
- **Session limits:** 30 concurrent database sessions
- **Workload type:** Autonomous Data Warehouse, Autonomous Transaction Processing

## Excluded Features

Autonomous Database for Developers supports all the features offered by a regular Autonomous 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 Database for Developers offers fewer resources and features than the regular Autonomous 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 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 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 Database; however, they will not support critical one-off patches.
- **Database Application Development and Developer Tools:** With Autonomous Database for Developers, you can use all the developer-related features and built-in tools that an Autonomous Database offers. See Develop Applications with Autonomous Database in *Tasks* and Service Monitoring in *Features* for complete details.

Autonomous 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.

# Compute Management and Billing in Autonomous Database on Dedicated Exadata Infrastructure

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

> **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.

## Compute Models in Autonomous Database on Dedicated Exadata Infrastructure

Autonomous Database on Dedicated Exadata Infrastructure offers two compute models while configuring your Autonomous 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 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, you cannot assign 3.5 ECPUs to a database. The next available number of ECPUs above 3 is 4.

  > **Note:**
  >
  > You can create Autonomous Database for Developers instances on ECPU based container databases. They are free Autonomous Databases that developers can use to build and test new applications. See Autonomous 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.

  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, you cannot assign 3.5 OCPUs to a database. The next available number of OCPUs above 3 is 4.

- 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.

You can choose a compute model while provisioning an Autonomous Exadata VM Cluster resource. The compute type selected for an Autonomous Exadata VM Cluster applies to all its Autonomous Container Databases and Autonomous Database instances.

## Compute Management in Autonomous Database

Autonomous 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 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 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:**
>
> 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 2 OCPUs or 8 ECPUs per node when created, these CPUs are not billed. Only once you provision Autonomous 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.

> **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, 2 OCPUs or 8 ECPUs 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 Database in that ACD, the new database consumes the initially allocated CPUs (2 OCPUs or 8 ECPUs per node). If the new database needs more than 2 OCPUs or 8 ECPUs, they get assigned from the parent AVMC's available CPUs, there by reducing the available CPUs at the parent

AVMC level. As you create more ACDs and provision Autonomous 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.

> **Note:**
>
> When you create an Autonomous 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 Create an Autonomous Container Database for instructions. These additional CPUs are not included in the billing.

Similarly, when you manually scale the CPUs of an Autonomous Database up, CPUs are consumed from the available CPUs at its parent AVMC level and its value changes accordingly.

**When an Autonomous 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 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 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.

> **Note:**
>
> 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.

## CPU Allocation When Auto-Scaling

The auto-scaling feature enables an Autonomous 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. See CPU Overprovisioning for more details.

To ensure that no single Autonomous Database can auto-scale up to consume all CPUs available in the pool for the overall deployment, Oracle Autonomous Database on Dedicated Exadata Infrastructure uses the Autonomous Container Database as a limiting control.

While provisioning an auto-scaling enabled Autonomous 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 Databases in this ACD.

When automatically scaling up an Autonomous Database, Oracle Autonomous Database on Dedicated Exadata Infrastructure looks for idle CPUs in its parent container database. If idle CPUs are available, the Autonomous 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 Database came from another running Autonomous Database that is lightly loaded and so not is using all of its allocated CPUs, Oracle Autonomous 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 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 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 Databases, all with auto-scaling enabled, and one stopped 8-CPU Autonomous 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 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.

## CPU Billing Details

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

1. CPU usage for each Autonomous Database is measured each second in units of whole OCPU or 4 ECPUs. A stopped Autonomous Database uses zero OCPU or ECPU. A runningAutonomous Database uses its allocated number of OCPUs or ECPUs plus any additional OCPUs or ECPUs due to auto-scaling.
2. The per-second measurements are averaged across each hour interval for each Autonomous Database.
3. The per-hour averages for the Autonomous Databases are added together to determine the CPU usage per hour across the entire Exadata Infrastructure resource.

## How Compartment Quotas Affect CPU Management

Normally, when you create or scale up an Autonomous Database, the ability of Oracle Autonomous 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 databases of each workload type (Data Warehouse or 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 databases in the VM cluster nodes, and the consequences of such placement on auto-scaling and parallel processing.

- **When you create an Autonomous Database**, its CPU allocation is split across multiple VM cluster nodes if the CPU count you specify is greater than the database split threshold defined at the ACD level, or the number of CPUs per node for the AVMC, whichever is smaller.

  For example, suppose you created an AVMC resource with two nodes and 5 OCPUs per node and you created an ACD in this AVMC with the database split threshold set to 16. As 5 is smaller than 16, any Autonomous Database with a CPU requirement greater than 5 will be split and opened across multiple nodes, allowing DML requests across those nodes. However, if the AVMC was created with two nodes and 20 OCPUs per node, any database with an OCPU requirement greater than 16 will be split and opened across multiple nodes.
- **When you manually scale an Autonomous Database**, its CPU allocation will be rebalanced to match this same allocation model. Single VM cluster node is used for CPU counts less than or equal to the database split threshold set at the ACD level, or the CPU count per node of the AVMC resource, whichever is smaller; split across multiple VM cluster nodes for CPU counts greater than the database split

threshold set at the ACD level, or the CPU count per node of the AVMC resource, whichever is smaller.

How an Autonomous 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 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 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 Database is called **provisionable CPUs**.

When you try to provision or scale an Autonomous 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 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 Database. See GetAutonomousDatabase for more details.

# Get Help and Contact Oracle Support

When using Autonomous 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.

Chances are that someone else has likely asked your question in the past. To find out, use scoped search to look for answers in our documentation and our forum platforms. To perform a scoped search, go to your favorite search engine and specify the site URLs along with your specific search terms, as follows:

**<*Your Search Terms*> site:oracle.com**

If you don't find the answer, you can remove **site:oracle.com** to see whether one of the many public forums whose members discuss Oracle Cloud and Oracle Database give you the help you need.

If you can't find an answer to your question through search, 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 Support

If you are having a technical issue using Autonomous Database on dedicated infrastructure, you can contact Oracle Support by creating a service request in My Oracle Support.

> **Note:**
>
> If you have not created a service request before, see Set Up Access to My Oracle Support to perform any necessary one-time activities to access My Oracle Support.

> **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.

1. Go to the My Oracle Support website at support.oracle.com and sign in with your Oracle account.
2. If you see "MY ORACLE SUPPORT" next to the Oracle logo at the top of the page, click **Switch to Cloud Support**.
3. Click **Service Requests** and then, after the page refreshes, click **Create Technical SR**.

   The Create Technical Service Request wizard is displayed.
4. For **Service Type**, select **Autonomous Database Dedicated**.

   The **Service Name** and **Problem Type** fields are displayed after **Service Type**.
5. For **Service Name**, select the value corresponding to your tenancy's CSI.
6. For **Problem Type**, locate and select the problem type that most fits your issue. For example, you might select Database Administration > General DB Admin Issues.

   A list of information you need to provide for the problem type you selected appears after the **Description** field.
7. For **Problem Summary**, enter a brief statement (up to 100 characters) describing your issue.
8. For **Description**, enter a description (up to 2000 characters) of what went wrong, what you were doing that led up to it, and what you expected to happen.
9. Complete the list of required information that follows the **Description** field.

   In addition to information about when you ran into your problem, you are often asked to provide the name and OCID (Oracle Cloud Identifier) of your tenancy and the cloud resource or resources you are having a problem with.

To get information about your tenancy, sign into your Oracle Cloud account, click the
profile icon in the top corner to open the Profile menu, and then click the link for your tenancy to display a page that shows your tenancy's name, OCID and other information.

To get similar information for a cloud resource (like an Autonomous Database), go to the details page for that resource.

10. Check that your contact information is correct, and then click **Next** to go to the **Additional Details** step.
11. At the **Additional Details** step, provide additional information about your issue, if requested, and upload any attachments that will make troubleshooting your issue easier. Then, click **Next** to go to the **Confirmation** step.
12. At the **Confirmation** step, check the information you have provided. If you need to change anything, click **Back** to go back and change information. When you are satisified, click **Submit**.

    The Create Technical Service Request wizard closes and your newly created service request is displayed in the list of your Technical SRs.

# Set Up Access to My Oracle Support

To create service requests with My Oracle Support, you must first perform these one-time activities:

- Get (or have) an Oracle account, sometimes called an oracle.com or Oracle SSO account. This account, different from your Cloud account, gives you access to a variety of services on the Oracle website, like making purchases through Oracle Store and logging service requests in My Oracle Support.

  If you don't already have an Oracle account, see Get an Oracle Account for instructions.
- Use your Oracle account to register your tenancy's Customer Support Identifier (CSI) in My Oracle Support.

  See Register Your CSI Number in My Oracle Support for instructions.
- Link your Oracle account to your Cloud account in Oracle Cloud.

  See Link Your Oracle Account to Your Cloud Account for instructions.

### Get an Oracle Account

1. Go to the Oracle website at www.oracle.com.
2. Click the
   View Accounts icon. Then, under Oracle Account, click **Create an Account**.

   The Create Your Oracle Account page is displayed.
3. Enter your email address and other details in the appropriate fields. Be sure to complete all the required fields.
4. Click **Create Account**.

   After your account is created, you'll receive a confirmation email at the email address that you provided.
5. Follow the instructions in the email to verify your email address.

### Register Your CSI Number in My Oracle Support

1. Get the CSI (Customer Support Identifier) number for your tenancy:
   a. Sign in to your Oracle Cloud account at cloud.oracle.com.
   b. Click the profile icon in the top corner to open the Profile menu. Then, click the link for your tenancy.

      A page showing information about your tenancy is displayed.
   c. In the **Tenancy Information** tab, locate the **CSI Number** field and note down your tenancy's CSI number.
2. Go to the My Oracle Support website at support.oracle.com and sign in with your Oracle Account.
3. If prompted, accept the My Oracle Support terms of use.
4. From the logged-in user menu, click **My Account**.
5. In the **Support Identifiers** section, click **Request Access**. The **Support Identifiers** section displays the accounts that your user name is currently associated with.
6. Enter your CSI number, an optional note to approver, and then click **Request Access**.
7. Enter your company name that owns the CSI number and click **Validate**.
8. Click **Next** and enter your contact details.
9. Accept the terms and conditions and click **Next**.

The status of your request is pending until you receive approval from the Customer User Administrator (CUA) for your tenancy's CSI.

### Link Your Oracle Account to Your Cloud Account

1. Sign in to your Oracle Cloud account at cloud.oracle.com.
2. Click the icon in the top corner to open the navigation menu. Under **Governance and Administration**, go to **Identity** and click **Users**.

   A list of the users in your tenancy is displayed.
3. Find and click your user name.

   Your user details are displayed.
4. Click **Link Support Account**.

   The Oracle account sign in page prompts you to enter your Oracle credentials.
5. Enter the user name and password of the Oracle account that you used to register your CSI number in My Oracle Support, and then click **Sign in**.

   Your Cloud account is linked to the Oracle account. The e-mail address associated with the Oracle account is displayed in the user details in the field **My Oracle Support account**.

# 2
# Quick Start

## Before You Begin with Autonomous Database

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

Before you create Autonomous Database on Dedicated Exadata Infrastructure deployments:

- On Oracle Cloud, purchase a subscription. You cannot create an Autonomous Database on Dedicated Exadata Infrastructure deployment until you do so.
- Optionally, if you want to leverage an object store for data loading, you need your object store credentials to use with Autonomous Database on Dedicated Exadata Infrastructure, including a *username* and a *password*. For details on the required credentials depending on the object store you want to use, see the following:
  - **Oracle Cloud Infrastructure Object Storage**:
    - `username` is your Oracle Cloud Infrastructure user name.
    - `password` is your auth token. See Working with Auth Tokens for more details.
  - **Oracle Cloud Infrastructure Object Storage Classic**:
    - `username` is your Oracle Cloud Infrastructure Classic user name.
    - `password` is your Oracle Cloud Infrastructure Classic password.

After you create Autonomous Database on Dedicated Exadata Infrastructure deployments:

- Optionally, you can reset the administrator password after you create an Autonomous Database on Dedicated Exadata Infrastructure deployment. While creating an Autonomous Database on Dedicated Exadata Infrastructure deployment, you are required to set the administrator password. If you want to change the administrator password or if you need to unlock the administrator account, see Unlock or Change the ADMIN Database User Password.

## Workflows

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

To get started with Autonomous 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 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.<br>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:<br>• Service Limits in *Oracle Cloud Infrastructure Documentation*<br>• Characteristics of Infrastructure Shapes<br>• Resource Limits<br>• Create an Exadata Infrastructure Resource<br>• Create an Autonomous Exadata VM Cluster<br>Deploying on Exadata Cloud@Customer:<br>• Preparing for Exadata Cloud@Customer<br>• Provisioning Exadata Cloud@Customer Systems<br>• Create an Autonomous Exadata VM Cluster |
| 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 Databases | Create at least one Autonomous Database in an Autonomous Container Database resource. | Create a Dedicated Autonomous Database |

# Create and Use Autonomous Database

You can access your Autonomous 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 Databasewith 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 Database | Create an Autonomous Database instance. | Create a Dedicated Autonomous Database |
| Start the instance | Start and verify that an Autonomous Database is up and running. | Start a Dedicated Autonomous Database |
| Create database users and obtain security credentials | Create accounts for your users and assign them appropriate privileges. Assign the necessary Autonomous Database roles. | Create Database Users |

| Task | Description | More Information |
|---|---|---|
| Connect to your database | Obtain security credentials and connect to your Autonomous Database using a database client or tool. | Connect to a Dedicated Autonomous 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 Database |
| Load data into the database | Use the available tools to load data into the database. | Loading Data into Autonomous Database |
| Monitor the database | Check on the health and performance of the database. | Manage and Monitor Performance |
| 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.<br>Back Up and Restore Autonomous Databases |

## Develop Applications with Autonomous Database

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

| Task | Description | More Information |
|---|---|---|
| Create an Autonomous 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 Database using a database client or tool. | Connect to a Dedicated Autonomous 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 Database. | The Basics of Building an Application |

# Access Autonomous Database in 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 Database** in the navigation menu.
6. Choose an item under Oracle Database based on whether your dedicated infrastructure resources are on Oracle Cloud or Oracle Exadata Cloud@Customer:
   - **On Oracle Cloud:** click **Autonomous Database** or **Autonomous Dedicated Infrastructure**.

     The **Autonomous Databases** page opens, showing the list of databases in your current Compartment.
   - **On Oracle Exadata Cloud@Customer:** click **Exadata Cloud@Customer**.

     The **VM Clusters** page opens. On this page, click the name of the kind of Autonomous Database resource you want to manage.

# LiveLabs for Autonomous Database on Dedicated Exadata Infrastructure

To try out Autonomous 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 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 Database workshops:

- Click Oracle LiveLabs, and search for *Autonomous Database* in the **Search Bar**.

  Oracle LiveLabs offers the following workshops for Autonomous Database, catering to different user roles.

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

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

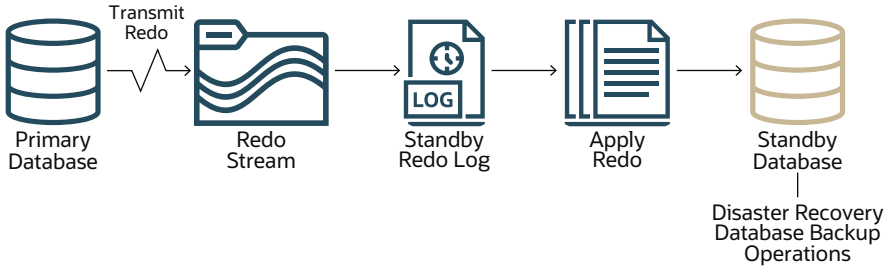
### About Autonomous Data Guard

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 automatically converts the standby database to the primary database and, as such, it begins servicing your applications.

The primary and standby databases are often called *peer* databases of each other.

> **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 the 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 Database from the database's Details page by choosing **Autonomous Data Guard Associations** under Resources in the side menu. You can view current lag values across all the Autonomous Databases in a container database from the container database's Details page in a similar fashion.

### Configuring Autonomous Data Guard

In Oracle Autonomous Database on Dedicated Exadata Infrastructure on dedicated infrastructure, you configure and manage Autonomous Data Guard at the Autonomous Container Database (ACD) level. You can enable Autonomous Data Guard for both new ACDs and already provisioned ACDs on Oracle Public Cloud deployments.

When you create an Autonomous Container Database, you select the **Enable Autonomous Data Guard** option and then provide details about the standby database, most notably the Exadata Infrastructure and Autonomous Exadata VM Cluster resources you want it created in and the protection mode you want to use. For information about the choices you have regarding these two details, see Autonomous Data Guard Configuration Options.

You can enable Autonomous Data Guard and add a standby ACD to an existing ACD from its **Details** page on the Oracle Cloud Infrastructure (OCI) console. However, you cannot enable Autonomous Data Guard on an ACD with an active maintenance run scheduled within the next two weeks. 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. See Enable Autonomous Data Guard on an Autonomous Container Database for instructions.

### Role Transitions and Operations

After an Autonomous Container Database is created, you can change the role of the peer databases using either a switchover or a failover operation. Additionally, should the primary database become unavailable for any reason, Autonomous Data Guard automatically performs a failover operation.

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. In general, you perform a switchover operation to restore the roles of peer databases to their original configuration after a failed primary database has been reinstated as a standby. 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. In general, you do not need to perform a failover operation manually. However, in the rare case when an automatic failure operation fails, 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.

When a failed primary database becomes available again, its role is set to **Disabled Standby**. At this point, you re-enable it by performing a **reinstate** operation. To perform a reinstate operation, see Reinstate the Disabled Standby in an Autonomous Data Guard Configuration.

#### Automatic Failover or Fast-Start Failover

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

Automatic failover is not enabled by default. You can enable automatic failover by selecting the **Enable automatic failover** option, while configuring the Autonomous Data Guard. See Update Autonomous Data Guard Settings to enable or disable automatic failover.

> **Note:**
>
> You can not enable Automatic Failover for databases with cross-region Autonomous Data Guard setup on Exadata Cloud@Customer deployments.

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 configured data loss guarantee can be upheld. You can modify the Fast Start failover lag limit to any value between 5 and 3600. See Update Autonomous Data Guard Settings for instructions.

  > **Note:**
  >
  > Minimum fast start lag limit value supported by ACDs with 19.17 and older versions is 30 seconds.

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.

### 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.

> **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 Databases in the snapshot standby ACD are updated. To access the database, reload the wallets from standby Autonomous 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 Databases
- Scale up or scale down Autonomous Databases
- Restore Autonomous 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 only 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 Databases.

#### 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. 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 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 Associations** under Resources in the side menu.

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 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 create an Autonomous Container Database with Autonomous Data Guard enabled, 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 three 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.

### About Protection Modes

Autonomous Data Guard provides these data protection modes:

- **Maximum Availability.** This protection mode provides the highest level of data 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*.

## 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 of primary and standby container databases separately and independently, just as though they were standard container databases.
- **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 databases.
  - Terminating a **standby container database** terminates the standby container database and removes the Autonomous Data Guard configuration from the primary container database, making it a standard 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

# Backup and Restore Autonomous Database on Dedicated Exadata Infrastructure

To support high availability, Autonomous 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.

Autonomous Database supports both automatic and manual backups. It even lets you create long-term backups that are retained beyond the backup retention policy defined at an ACD level. 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. Terminating an Autonomous Database deletes all of its automatic and manual backups, including long-term backups.

## About Backup and Recovery

**Supported Backup Types**

- **Automatic Backups:** By default, Autonomous 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.

  > **Note:**
  >
  > You can not disable automatic backups while provisioning an ACD with Autonomous Data Guard.

  Once the automatic backups are enabled you can not disable them for an ACD.

- **Manual Backups:**
  - In addition to the backups created by Autonomous Database automatically, you can also create **on-demand manual backups** from the Oracle Cloud Infrastructure (OCI)

console; for example if you want to take a backup before a major change to make restore and recovery faster. These backups are retained *up to 95 days* depending on the backup retention policy (period) chosen for the ACD. Manual backups are cumulative incremental backups containing all the incremental backups from the last full backup.

- Apart from the regular on-demand backups, you can also create **long-term backups** for an Autonomous Database from its details page on the OCI console. Long-term backups are retained beyond the backup retention policy (period) defined at the ACD level. While creating a long-term backup, you can specify its retention period anywhere from *90 days to 10 years.*

> **Note:**
>
> Long-term backups cannot be used for Autonomous Database in-place restore operations. They are exclusively intended for cloning into new Autonomous Databases.

**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.

On Oracle Public Cloud deployments, 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*.

**Backup Destination**

APPLIES TO: Exadata Cloud@Customer only

While provisioning an ACD or while enabling automatic backups on an already created ACD, you can configure the backup destination for backups of Autonomous Databases created in it. As of the current release, the backup destination type can only be set while enabling automatic backups on an ACD and can not be changed later. Below listed are the different backup destination options for Autonomous Database on Exadata Cloud@Customer:

- **Object Storage**: Stores backups in an Oracle-managed object storage container on Oracle Cloud Infrastructure. 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.
- **Network File System (NFS)**: Stores backups in a Network File System (NFS) storage location. 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 Network File System

(NFS) as the type, select a previously defined backup destination that uses Network File System (NFS) storage.

– **Recovery Appliance**: 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.

> **Note:**
>
> Please ensure to provide the connection string that connects to the recovery appliance in an Oracle "easy connect" string format, that is, `<host>:<port>/<service name>`, where `<host>` is the SCAN hostname of the Zero Data Loss Recovery Appliance.

**Restore and Recover**

You can either restore the database from a specific backup or use a timestamp for point-in-time recovery 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.

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 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 Database Manually |

| Operation | Details | Further Reference |
|---|---|---|
| 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 Database.<br>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 Database on Dedicated Exadata Infrastructure<br>For instructions on cloning a manual or long-term backup from the OCI console, see Clone Autonomous Database from Backup. |
| 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:<br>• Backup retention period to any value between 7 to 95 days.<br>• Backup destination type (applies to Exadata Cloud@Customer only.<br>• HTTP Proxy. This is applicable only when Object Storage is selected as the backup destination for Exadata Cloud@Customer deployments. | 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 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 Database |

## 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 Database on Dedicated Exadata Infrastructure

# High Performance

## High Performance Features in Autonomous Database on Dedicated Exadata Infrastructure

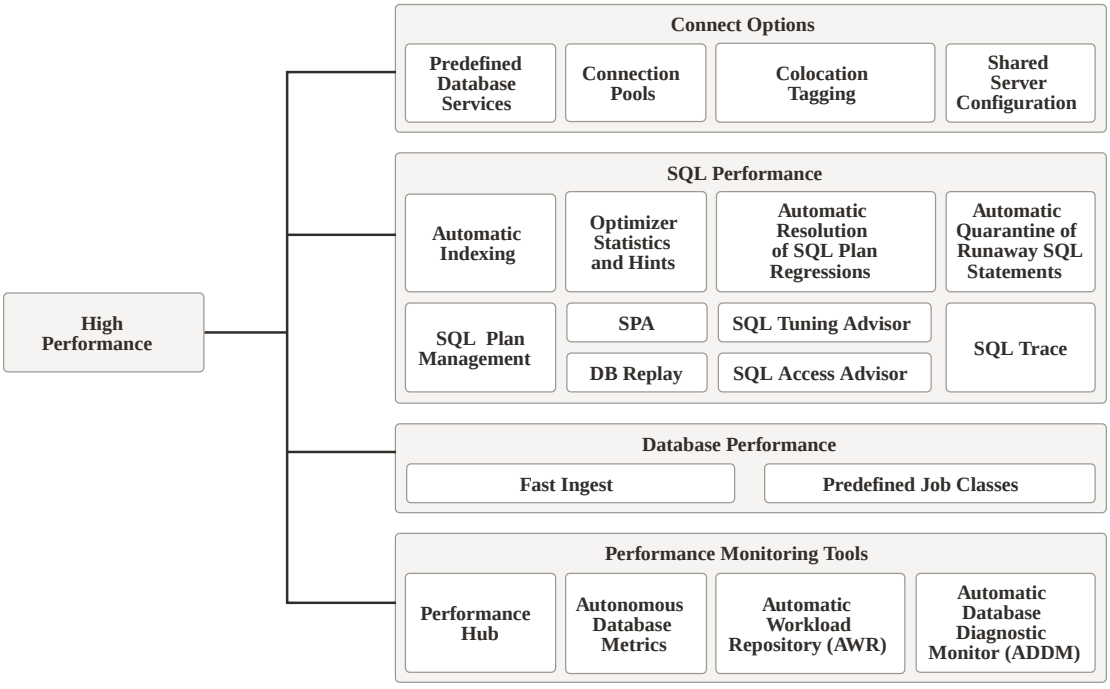
This article describes the key performance tuning features in Autonomous Database on Dedicated Exadata Infrastructure.

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 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 Database, and instructions on how to use them, see Oracle Database SQL Tuning Guide.

You can see a broad categorization for the key performance tuning features of Autonomous 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 Database, the performance of your application's interaction with the database depends on which database service you connect to. Autonomous 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 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 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 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.

- **Shared Server Configuration** is another feature Autonomous 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 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 Database created under a Shared Server-enabled Autonomous Container Database, and you can not use a Dedicated connection for Autonomous Databases created under a Shared Server-enabled Autonomous Container Database.

## SQL Performance Tuning Features

Great applications begin with well written SQL. Oracle Autonomous 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 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 Database to achieve this.

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

## Optimizer Statistics

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

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

## 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 Database defaults for the optimizer and `PARALLEL` hints differ depending on your workload:

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

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

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

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

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

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

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

- **Autonomous Transaction Processing:** Autonomous 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 Database. Auto indexing is disabled by default in Autonomous Database.

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

The automatic indexing feature in Autonomous 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 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 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 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 *Database Performance Tuning Guide*.

To use fast ingest with your Autonomous 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 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 Database 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 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 Database.

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

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

Notes for Oracle Scheduler:

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

See Scheduling Jobs with Oracle Scheduler for more information on Oracle Scheduler and `DBMS_SCHEDULER.CREATE_JOB`.

See SET_ATTRIBUTE Procedure for information on job attributes.

## 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, Oracle Enterprise Manager and Oracle Management Cloud.<br>See Monitor Database Performance with Performance Hub for more details. |
| Autonomous Database Metrics | The Autonomous 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 Database resources.<br>See Monitor Databases with Autonomous Database Metrics for more information such as it prerequisites, usage, and the list of metrics available for Autonomous 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).<br>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.<br>For instructions on using these tools, as well as detailed information about database performance monitoring and tuning, see Oracle Database Performance Tuning Guide.<br>AWR and ADDM are also available from Performance Hub. See Monitor Database Performance with Performance Hub for more details.<br>For a quick introduction to database performance monitoring and tuning, see Oracle Database 2 Day + Performance Tuning Guide. |

# Use SQL Tracing on Autonomous Database on Dedicated Exadata Infrastructure

You can use SQL Tracing with Autonomous 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.

> **Note:**
>
> You can use SQL Tracing only with Autonomous Database versions 19.18 or later.

### 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 Database to achieve this.

SQL tracing is disabled by default in Autonomous 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 Database for more information.
- Then, enable SQL tracing. See Enable SQL Tracing on Autonomous 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 Database for more information.
  - View and analyze SQL Trace data saved to the view `SESSION_CLOUD_TRACE`. See View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database for more information.

## Configure SQL Tracing on Autonomous Database

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

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

   To save the SQL tracing files, the bucket can be in any Cloud Object Store that Autonomous Database supports. For example, to create a bucket in Oracle Cloud Infrastructure Object Storage, 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 Database supports.

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

For example:

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

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

## Enable SQL Tracing on Autonomous Database

> **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 Database

You use SQL trace file data to analyze application performance on Autonomous Database. When you disable SQL trace in your database session, data is written to the Cloud Object Store bucket configured with `DEFAULT_LOGGING_BUCKET`.
The SQL Trace facility writes the trace data collected in the session to Cloud Object Store in the following format:

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

The components of the file name are:

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

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

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

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

```
sqltrace/sqlt_test/modname/sqltrace_5415_56432.trc
```

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

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

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

   The input and output files are the only required arguments.
4. To view online Help, invoke TKPROF without arguments.

   See *Tools for End-to-End Application Tracing* in Oracle Database SQL Tuning Guide for information about using the TKPROF utility.

## View Trace Data in SESSION_CLOUD_TRACE View on Autonomous Database

When you enable SQL Tracing, the same trace information that is saved to the trace file on Cloud Object Store is available in the SESSION_CLOUD_TRACE view in the session where the tracing was enabled.
While you are still in the database session you can view SQL tracing data in the SESSION_CLOUD_TRACE view. The SESSION_CLOUD_TRACE view includes two columns: ROW_NUMBER and TRACE.

```
DESC SESSION_CLOUD_TRACE

Name        Null? Type
----------- ----- ------------------------------
ROW_NUMBER        NUMBER
TRACE             VARCHAR2(32767)
```

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

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

For example:

```
SELECT trace FROM SESSION_CLOUD_TRACE 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 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 Databases, 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 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.

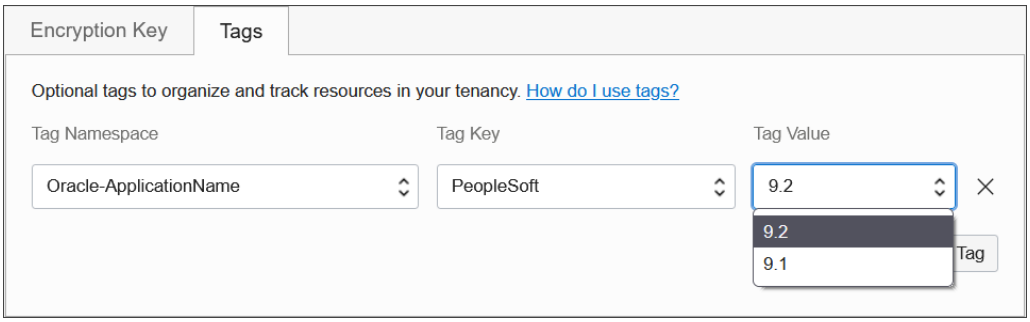
There are many cloud resources that can be tagged in an Autonomous Database deployment. Exadata Infrastructure, Autonomous Exadata VM Cluster, Autonomous Container Database, and Autonomous Datatabase are some of them. Tags can either be applied while creating the resources or modified later. For example, 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.(Required) Enter introductory text here, including the definition and purpose of the concept.

> **Tip:**
>
> For a "try it out" tutorial that demonstrates implementing tags in Oracle Autonomous Database, refer to *Lab 14: Oracle Standard Tags* in Oracle Autonomous 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 Database, can ensure that the database is configured ready for the particular application (Example: Peoplesoft) out of the box.




## 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**.

| Tag Key | Tag Value Options | Description |
|---|---|---|
| OracleStandard.Criticality | Extreme<br>High<br>Medium<br>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.<br>For example, a database resource with *OracleStandard.Criticality* set to *Extreme* or *High* may require the highest availability SLA, and may need to be configured with Autonomous Data Guard. |
| OracleStandard.Environment | Dev<br>Test<br>Prod<br>Pre-prod<br>Staging<br>Trial<br>Sandbox<br>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<br>Internal<br>Sensitive<br>Highly Sensitive<br>Extremely Sensitive | An application or database classification tag. *OracleStandard.Sensitivity* set to *Highly Sensitive* may indicate that an access control list or certain Network Security Group (NSG) enforcement is mandatory to restrict access. |

| Tag Key | Tag Value Options | Description |
|---|---|---|
| OracleStandard.Regulation | Refer to List of Compliance Regulations for values. | Denotes one or more compliance regulations that a resource must adhere to.<br>Tag administrators may add values to the list from the OCI *Governance and Administration* console. Refer to Using Predefined Values for more details. |
| OracleStandard.TargetUsers | Public<br>Customers<br>Partners<br>Company<br>Division<br>Department<br>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<br>10<br>100<br>1000<br>10000<br>100000<br>1000000<br>1000000<br>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. |
| 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.<br>Tag administrators may add relevant values to the list from the OCI *Governance and Administration* 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 |
| 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<br>11.1 | Denotes the version of the Hyperion application. |
| JD Edwards | 9.2<br>9.1<br>9.0 | Denotes the version of the JD Edwards application. |

| Tag Key | Tag Value Options | Description |
|---|---|---|
| Oracle_E-Business_Suite | 12.2<br>12.1<br>12.0<br>11i | Denotes the version of the Oracle E-Business Suite application. |
| PeopleSoft | 9.2<br>9.1 | Denotes the version of the Peoplesoft application. |
| Siebel | 8.2<br>8.1 | Denotes the version of the Siebel application. |
| 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 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:

- OCPUs available to dedicated deployments of Autonomous Data Warehouse databases.
- ECPUs available to dedicated deployments of Autonomous Data Warehouse databases.
- Total data storage available to dedicated deployments of Autonomous Data Warehouse databases.
- OCPUs available to dedicated deployments of Autonomous Transaction Processing databases.
- ECPUs available to dedicated deployments of Autonomous Transaction Processing databases.
- Total data storage available to dedicated deployments of Autonomous 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-ocpu-count` quota to 3 OCPUs in the compartment `MyCompartment` :

```
set database quota atp-dedicated-ocpu-count to 3 in compartment
MyCompartment
```

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

```
unset database quota atp-dedicated-ocpu-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 Database on Oracle Public Cloud

Family name: `database`

| Name | Scope | Description |
|---|---|---|
| `adw-dedicated-ocpu-count` | Availability domain | Number of Autonomous Data Warehouse 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-ecpu-count` | Availability domain | Number of Autonomous Data Warehouse ECPUs 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 Data Warehouse databases using dedicated Exadata infrastructure. (See note following this table about "n/a" values on the **Limits, Quotas and Usage** page of the Console.) |

| Name | Scope | Description |
|---|---|---|
| `atp-dedicated-ocpu-count` | Availability domain | Number of Autonomous 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-ecpu-count` | Availability domain | Number of Autonomous 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-total-storage-tb` | Availability domain | Amount of storage (in TB) for Autonomous 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 Database on Exadata Cloud@Customer

Family name: `database`

| Name | Scope | Description |
|---|---|---|
| `adw-exac@c-ocpu-count` | Availability domain | Number of Autonomous Data Warehouse OCPUs for databases using Exadata Cloud@Customer. (See note about "n/a" values on the Limits, Quotas and Usage page of the Console.) |
| `adw-exac@c-ecpu-count` | Availability domain | Number of Autonomous Data Warehouse ECPUs 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-exac@c-total-storage-tb | Availability domain | Amount of storage (in TB) for Autonomous Data Warehouse 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-exac@c-ocpu-count | Availability domain | Number of Autonomous 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-exac@c-ecpu-count | Availability domain | Number of Autonomous 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-exac@c-total-storage-tb | Availability domain | Amount of storage (in TB) for Autonomous 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 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 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 Database on Dedicated Exadata Infrastructure offers several pairs of predefined database services to choose from, as described in Predefined Database Service Names for Autonomous 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 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 `KEEP`s 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 oraccess.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 your 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('HIGH');
```

To enable Application Continuity for a service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_AC('TPURGENT');
```

To enable TAF SELECT for a service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAF('LOW');
```

To enable TAF BASIC for a service:

```
execute DBMS_APP_CONT_ADMIN.ENABLE_TAF('MEDIUM', 'SESSION');
```

To disable failover for a service:

```
execute DBMS_APP_CONT_ADMIN.DISABLE_FAILOVER('HIGH');
```

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('HIGH', 600, 'IMMEDIATE' );
```

To adjust only drain timeout :

```
execute DBMS_APP_CONT_ADMIN.SET_DRAINING('TPURGENT', 600);
```

## Configure Clients for FAN Including Optional Wallets

Wallet-based authentication is an option for FAN when using Autonomous 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>
```

# Change Management

## Service Maintenance of Autonomous Database on Dedicated Exadata Infrastructure

Oracle schedules and performs all patching and other maintenance operations on all the Autonomous 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.

> **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 Database. These maintenance events vary in their scope and frequency of patching.

- **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 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 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 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 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.

- **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.

## 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 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 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.<br>• View and Manage Scheduled Maintenance of an Exadata Infrastructure Resource<br>• 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.<br>• View and Manage Scheduled Maintenance of an Autonomous Exadata VM Cluster<br>• View Past Maintenance of an Autonomous Exadata VM Cluster<br><br>**Note:**<br>AVMC resources provisioned on the Exadata Infrastructure resources in Oracle Cloud before the launch of the multiple VM Autonomous Databases 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:<br>– **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.<br><br>**Note:**<br>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.<br><br>– **Maintenance version** for the automatic updates (**Next RU** or **Latest RU)**.<br>– **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.<br><br>**Note:**<br>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.<br><br>• 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. |

| Infrastructure Resource | Notes & Further Reference |
| --- | --- |
| | **Note:**<br>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.<br>• View and Manage Scheduled Maintenance of an Autonomous Container Database<br>• 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

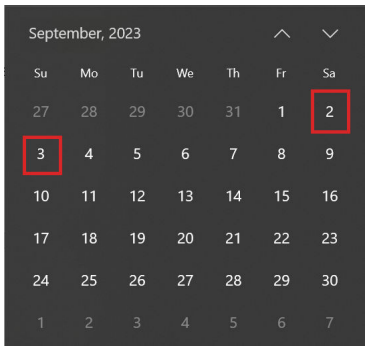
You can choose the following details from the Oracle Cloud Infrastructure console while defining a custom schedule for any of the above infrastructure resources.

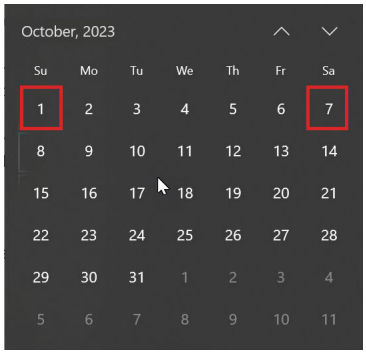
- **Permitted Months**: You must select at least one month per quarter.
- **Week (or Weeks) Within the Selected Months:** Weeks start on the 1st, 8th, 15th, and 22nd days of the month and have a duration of 7 days. Weeks start and end based on calendar dates, not days of the week. Maintenance cannot 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:




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.

**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 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 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 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

## Automatic Queuing of Maintenance Events

### Quarterly Maintenance Events of Different Autonomous 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.<br>If the quarterly maintenance is subsequently rescheduled, then the monthly security maintenance will begin immediately.<br>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.<br>**Note:**<br>Any time before the scheduled monthly Exadata Infrastructure maintenance, you can reschedule it.<br>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 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 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.

# Service Monitoring

## Plan and Monitor Capacity

You can monitor and plan the compute and storage resources of your Autonomous Database on Dedicated Exadata Infrastructure for efficient usage and optimal billing.

Oracle Autonomous 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 Databases or scale existing Autonomous Databases.
- **Provisioned CPUs**: Total CPUs allocated for all the Autonomous Databases within an Autonomous Exadata VM Cluster (AVMC).
- **Reserved CPUs**: Total CPUs reserved to support auto-scaling of Autonomous Databases, Autonomous 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 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 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 Databases' use in an Autonomous Exadata VM Cluster (AVMC).
- **Used Storage (in TBs):** Storage currently used by Autonomous 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 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 Databases that you can create. The smallest Autonomous Database can be created with 2 ECPUs or 0.1 CPUs. Therefore the maximum number of **Autonomous 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 Databases per Autonomous Container Database | 200 |
| Autonomous Databases per Autonomous Container Database with Autonomous Data Guard Configured | 25 |

> **Note:**
>
> It is possible to provision more Autonomous 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 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 per node are 5 OCPUs or 20 ECPUs, 100GB Memory, and 334GB Local Storage. You also need at least 2.87TB Exadata Storage per AVMC. Similarly, the minimum resources needed per node to create an ACD are 2 OCPUs or 8 ECPUs 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:**
>
> Database memory per ECPU (GB) and Database storage (TB) are set to 3GB and 2TB, respectively.

| Property | 1 ACD | 2 ACDs | 3 ACDs | 16 ACDs |
|---|---|---|---|---|
| ECPU Count | 40 | 40 | 48 | 256 |
| Memory (GB) | 200 | 200 | 224 | 848 |
| Local Storage (GB) | 677 | 780 | 883 | 2222 |
| Exadata Storage (TB) | 2.87 | 2.99 | 3.11 | 4.69 |

## 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 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 Databases in an AVMC. As you provision, run, and terminate Autonomous Databases or provision, delete, and restart ACDs, the compute

and storage resources move into different categories as explained in Compute Management in Autonomous Database.

Tracking resource usage for an AVMC or ACD across tenancies is critical in planning the capacity for your Autonomous Database on Dedicated Exadata Infrastructure. To simplify resource usage tracking, Oracle Autonomous Database provides insights in graphical and tabular formats from the Oracle Cloud Infrastructure (OCI) console.

Autonomous 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.

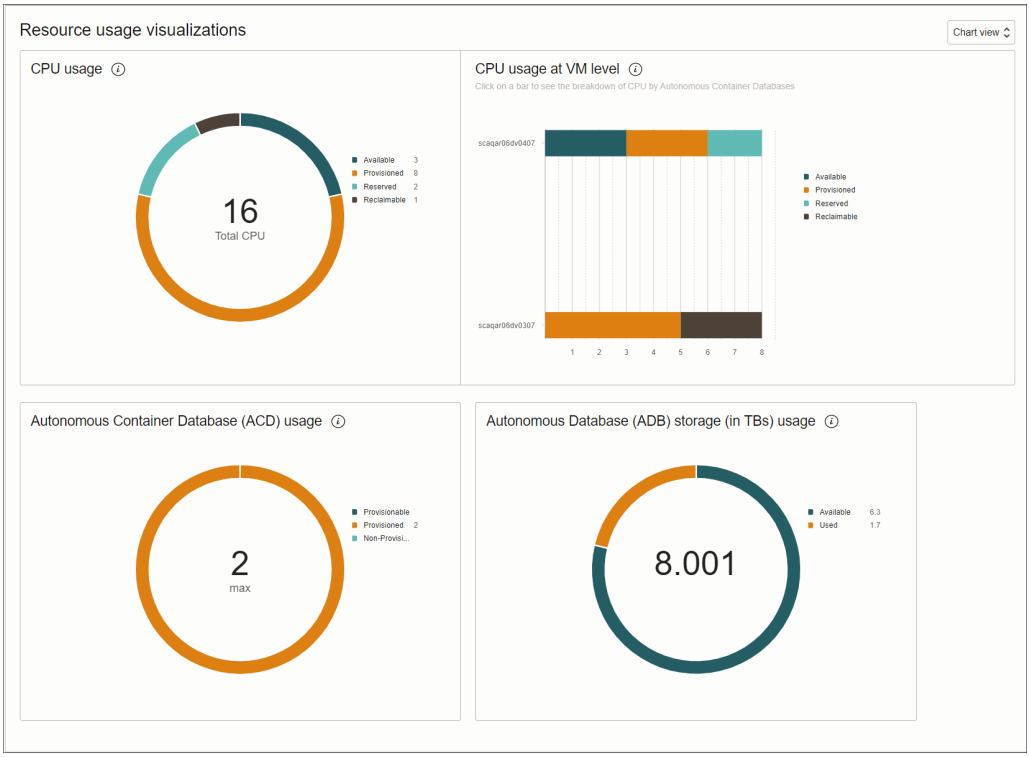
- Autonomous Exadata VM Cluster
- Autonomous Container Database

### 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:

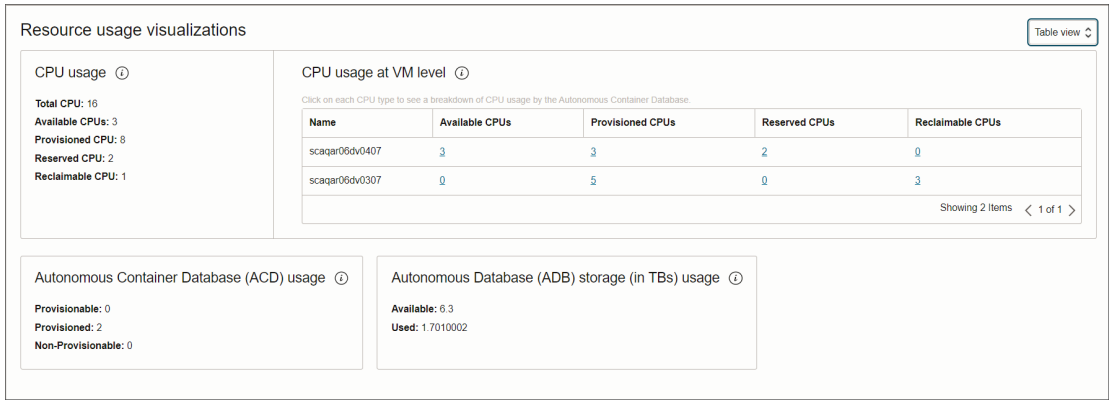



The four charts depict:

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 Database (ADB) storage (in TBs) usage:** This is a doughnut-chart depicting the available, used, and total Autonomous 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.




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 Database (ADB) storage (in TBs) usage:** Shows the available and used Autonomous Database storage in TBs.

## Autonomous Container Database

This section has graphical and tabular representations of the ACD's resource usage.

### Chart View

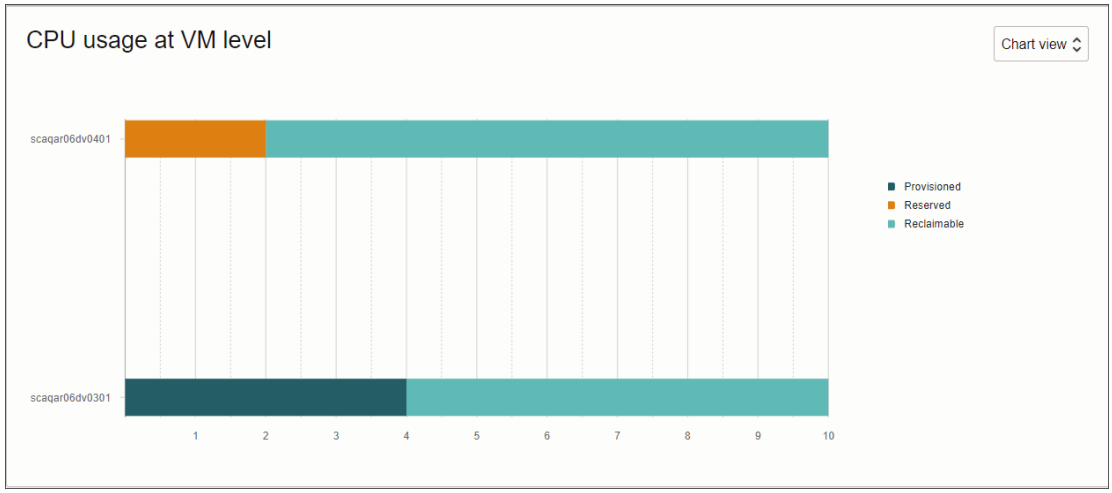
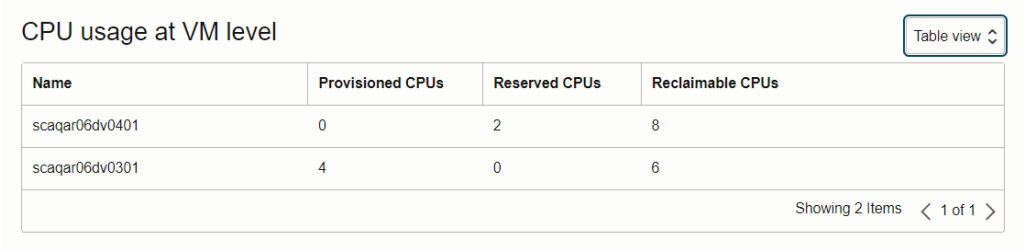



Chart view is the default view. In this view, you can see a horizontal bar graph giving a break-down of the CPUs used by this ACD in each VM. Each bar shows the number of provisioned, reserved, and reclaimable CPUs for that VM with color-coding. Hovering on each colored part of the horizontal bar displays the number of provisioned, reserved, and reclaimable CPUs for that specific VM.

**Table View**




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 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. All Shapes are based on Oracle Exadata X9M-2, X8M-2, X8-2, or X7-2 System Models.

The total resources allocated to your Autonomous 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 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:** 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.

# Monitor Databases with Autonomous Database Metrics

You can monitor the health, capacity, and performance of your Autonomous Databases with metrics, alarms, and notifications. You can use Oracle Cloud Infrastructure console or Monitoring APIs to view metrics.

The Autonomous 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 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 Database Metrics for a Database
- View Autonomous 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 Databases include the following **dimensions**:

- **AUTONOMOUSDBTYPE**
  The type of Autonomous Database, Autonomous Data Warehouse (ADW) or Autonomous Transaction Processing (ATP).
- **deploymentType**
  The Exadata infrastructure type, serverless or dedicated. When using the Console to view default metric charts for multiple Autonomous Databases, you must specify this dimension.
- **DISPLAYNAME**
  The friendly name of the Autonomous Database.
- **REGION**
  The **region** in which the Autonomous Database resides.
- **RESOURCEID**
  The **OCID** of the Autonomous Database.
- **RESOURCENAME**
  The name of the Autonomous Database.

The metrics listed in the following table are automatically available for any Autonomous Database that you create. You do not need to enable monitoring on the resource to get these metrics.

> **Note:**
>
> Valid alarm intervals are 5 minutes or greater due to the frequency at which these metrics are emitted. See To create an alarm for details on creating alarms.

In the following table, metrics that are marked with an asterisk (*) can be viewed only on the **Service Metrics** page of the Oracle Cloud Infrastructure console. All metrics can be filtered by the dimensions described in this topic.

| Metric | Metric Display Name | Unit | Description | Collection Frequency |
|---|---|---|---|---|
| `ApplyLag` | Apply Lag | seconds | This metric displays (in seconds) how far the standby database is behind the primary database as of the time sampled. | NA |
| `BlockChanges` | DB Block Changes | changes per second | The average number of blocks changed per second. | NA |
| `CpuTime*` | CPU Time | seconds per second | Average rate of accumulation of CPU time by foreground sessions in the database over the time interval.<br>Statistic: Mean<br>Interval: 1 minute | NA |
| `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.<br>Statistic: Mean<br>Interval: 1 minute | NA |
| `CurrentLogons*` | Current Logons | count | The number of successful logons during the selected interval.<br>Statistic: Count<br>Interval: 1 minute | 1 minute |

| Metric | Metric Display Name | Unit | Description | Collection Frequency |
|---|---|---|---|---|
| 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.<br>It is calculated as the average rate of accumulation of database time by foreground sessions in the database over the time interval.<br>Statistic: Mean<br>Interval: 1 minute | NA |
| ECPUsAllocated | ECPU Allocated | integer | The actual number of ECPUs allocated by the service during the selected interval of time. | NA |
| ExecuteCount | Execute Count | count | The number of user and recursive calls that executed SQL statements during the selected interval.<br>Statistic: Sum<br>Interval: 1 minute | NA |
| IOPS | IOPS | operations per second | The average number of I/O operations per second. | NA |
| IOThroughput | IO Throughput | MB per second | The average throughput in MB per second. | NA |

| Metric | Metric Display Name | Unit | Description | Collection Frequency |
|---|---|---|---|---|
| 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.<br>Statistic: Sum<br>Interval: 1 minute | NA |
| OCPUsAllocated | OCPU Allocated | integer | The actual number of OCPUs allocated by the service during the selected interval of time. | NA |
| ParsesByType | Parses By Type | parses per second | The number of hard or soft parses per second. | NA |
| ParseCount* | Parse Count (Total) | count | The number of hard and soft parses during the selected interval.<br>Statistic: Sum<br>Interval: 1 minute | 1 minute |
| QueuedStatements | Queued Statements | count | The number of queued SQL statements, aggregated across all consumer groups, during the selected interval.<br>Statistic: Sum<br>Interval: 1 minute | NA |
| RedoSize | Redo Generated | MB per second | The Average amount of redo generated in MB per second. | NA |
| RunningStatements | Running Statements | count | The number of running SQL statements, aggregated across all consumer groups, during the selected interval.<br>Statistic: Mean<br>Interval: 1 minute | 1 minute |

| Metric | Metric Display Name | Unit | Description | Collection Frequency |
|---|---|---|---|---|
| Sessions | Sessions | count | The number of sessions in the database.<br>Statistic: Sum<br>Interval: 1 minute | 1 minute |
| StorageAllocated* | Storage Space Allocated | GB | Maximum amount of space allocated to the database during the interval.<br>Statistic: Max<br>Interval: 1 hour | 1 hour |
| StorageAllocatedByTablespace* | Allocated Storage Space By Tablespace | GB | Maximum amount of space allocated for each tablespace during the interval. | NA |
| StorageUsed* | Maximum Storage Space | GB | Maximum amount of space used during the interval.<br>Statistic: Max<br>Interval: 1 hour | 1 hour |
| StorageUsedByTablespace* | Storage Space Used By Tablespace | GB | Maximum amount of space used by each tablespace during the interval. | NA |
| StorageUtilization | Storage Utilization | percent | The percentage of provisioned storage capacity currently in use.<br>Represents the total allocated space for all tablespaces.<br>Statistic: Mean<br>Interval: 1 hour | 1 hour |
| StorageUtilizationByTablespace* | Storage Space Utilization By Tablespace | percent | The percentage of space utilized by each tablespace. | NA |
| TransactionsByStatus | Transactions By Status | transactions per second | The number of committed or rolled back transactions per second. | NA |

| Metric | Metric Display Name | Unit | Description | Collection Frequency |
|---|---|---|---|---|
| `TransactionCount*` | Transaction Count | count | The combined number of user commits and user rollbacks during the selected interval.<br>Statistic: Sum<br>Interval: 1 minute | 1 minute |
| `TransportLag` | Transport Lag | seconds | The approximate number of seconds of redo not yet available on the standby database as of the time sampled. | 1 minute |
| `UserCalls*` | User Calls | count | The combined number of logons, parses, and execute calls during the selected interval.<br>Statistic: Sum<br>Interval: 1 minute | 1 minute |
| `WaitTime*` | Wait Time | seconds per second | Average rate of accumulation of non-idle wait time by foreground sessions in the database over the time interval.<br>Statistic: Mean<br>Interval: 1 minute | 1 minute |

# Monitor Database Performance with Performance Hub

You can use Performance Hub to monitor your databases for defined time periods and download statistical reports. Performance Hub lets you view real-time and historical performance data for an Autonomous Database.

> **Tip:**
>
> To try out Performance Hub, you can go through *Lab 10: Managing Database Performance* in the Oracle Autonomous Database Dedicated for Developers and Database Users Workshop.

You can access Performance Hub from the Autonomous Database **Details** page. To access Performance Hub:

1. Go to the **Details** page of the Autonomous Database you want to monitor with Performance Hub.

   For instructions, see View Details of a Dedicated Autonomous 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 Performance Hub to monitor. |
| 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.<br>You can hide or show the activity summary timeline by selecting or deselecting **Hide Activity Summary** located next to **Time Zone** above this timeline. |
| Reports | The **Reports** drop down menu is used to view the **Automatic Workload Repository** (AWR) report.<br>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.<br>An AWR report shows data captured between two points in time (or snapshots).<br>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.<br>The HTML report includes links that you can use to navigate quickly between sections. |
| 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.<br>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.<br>For more information, see ASH Analytics. |

| Name | Description |
|---|---|
| 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.<br>For more information, see SQL Monitoring. |
| Automatic Database Diagnostic Monitor (ADDM) | The **ADDM** tab provides access to analysis information gathered by the Automatic Database Diagnostic Monitor (ADDM) tool.<br>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.<br>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.<br>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.<br>This tab shows four chart areas that show database workload in various ways:<br>• **CPU Statistics**: Charts CPU usage.<br>• **Wait Time Statistics**: Shows the wait time across the database's foreground sessions, divided by wait classes.<br>• **Workload Profile**: Charts user (client) workload on the database.<br>• **Sessions**: Shows the number of sessions, successful logons, and current logons.<br>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.<br>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.

## Use Database Actions to Manage and Monitor Databases

> **Tip:**
>
> For a "try it out" alternative that demonstrates these instructions, run *Lab 7: Database Actions Console* in the Oracle Autonomous 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 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.

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 Monitor Databases

APPLIES TO: Oracle Public Cloud only

You can use Database Management Service to monitor the health of a single Autonomous Database or a fleet of Autonomous Databases.

Database Management Service lets you:

- Monitor the key performance and configuration metrics of a fleet of Autonomous Databases.
- Compare and analyze database metrics over a selected period.
- Group your critical Autonomous Databases, which reside across compartments, into a Database Group, and monitor them.

To enable Database Management Service:

- Perform the prerequisite steps as described in General Prerequisite Tasks.
- Obtain the required permissions as described in Permissions Required to Enable Database Management for Autonomous Databases.
- Obtain additional permissions to use the Database Management **Fleet Summary** and **Managed Database Details** pages and view alarms on the **Fleet Summary**

page. These are listed in Additional Permissions Required to Use Database Management.

- Finally, enable Database Management for Autonomous Databases by following the steps outlined in Enable Database Management for Autonomous Databases.

After enabling Database Management Service, you can perform the actions listed below.

| Action | Further Reference |
|---|---|
| Monitor the health of your fleet of Autonomous Databases on the **Fleet Summary** page. | Monitor the Health of Your Oracle Database Fleet<br>Assess the Performance of Your Databases at a Glance |
| Monitor a single Autonomous Database on the **Managed Database Details** page. | Monitor and Manage an Autonomous Database |
| Group Autonomous Databases that reside across compartments into a Database Group and monitor them. | Create and Use Database Groups |

> **Note:**
>
> Database Management features such as AWR Explorer and Performance Hub on Managed Database Details are *unavailable* for Autonomous Databases. However, Performance Hub for Autonomous Databases continues to be a free feature that you can access from the Autonomous Database Details page without enabling Database Management.
> For further information, see Use Performance Hub to Monitor Databases.

## Use Enterprise Manager to Manage and Monitor Databases

You can use Oracle Enterprise Manager to manage and monitor Autonomous 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.

> **Note:**
>
> Enterprise Manager version 13.3 and above support monitoring and management of Autonomous 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 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 Databases in *Enterprise Manager Cloud Control Administrator's Guide for Oracle Autonomous Databases*.

> **Tip:**
>
> For a "try it out" alternative that demonstrates how to connect to an Autonomous Database from Oracle Enterprise Manager, see *Lab 12: Deploy OEM and Connect ADB to OEM* in Oracle Autonomous 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 Databases*.

# Use Operations Insights on Autonomous Database

APPLIES TO: Oracle Public Cloud only

Operations Insights is a cloud-native service that enables users to make informed, data-driven, Autonomous Database resource and performance management decisions.

## About Operations Insights

Operations 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 Operations 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 Operations Insights with Autonomous Database, you must first enable it from the Oracle Cloud Infrastructure console. See Enable Operations Insights for a Dedicated Autonomous Database.

Operations 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 Operations Insights in *Oracle Cloud Infrastructure Documentation*.

## Use Work Requests to Monitor Long-Running Operations

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

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

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

## Events and Notifications to Watch Resource Activity

When Oracle Autonomous Database 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 Database event notifications through OCI Notification Service, see *Lab 11: OCI Notification Service* in Oracle Autonomous Database Dedicated for Fleet Administrators Workshop.

This combination of Autonomous Database events, the Events service and the Notification service provide a very rich set of options for keeping abreast of resource activity because:

- Autonomous 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 Database on Oracle Cloud, see Events for Autonomous Database on Dedicated Exadata Infrastructure.

  For listings of the operations that generate events for Autonomous 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.

# Security

## Security Features in Autonomous Database on Dedicated Exadata Infrastructure

This article describes the key security features in Autonomous Database on Dedicated Exadata Infrastructure.

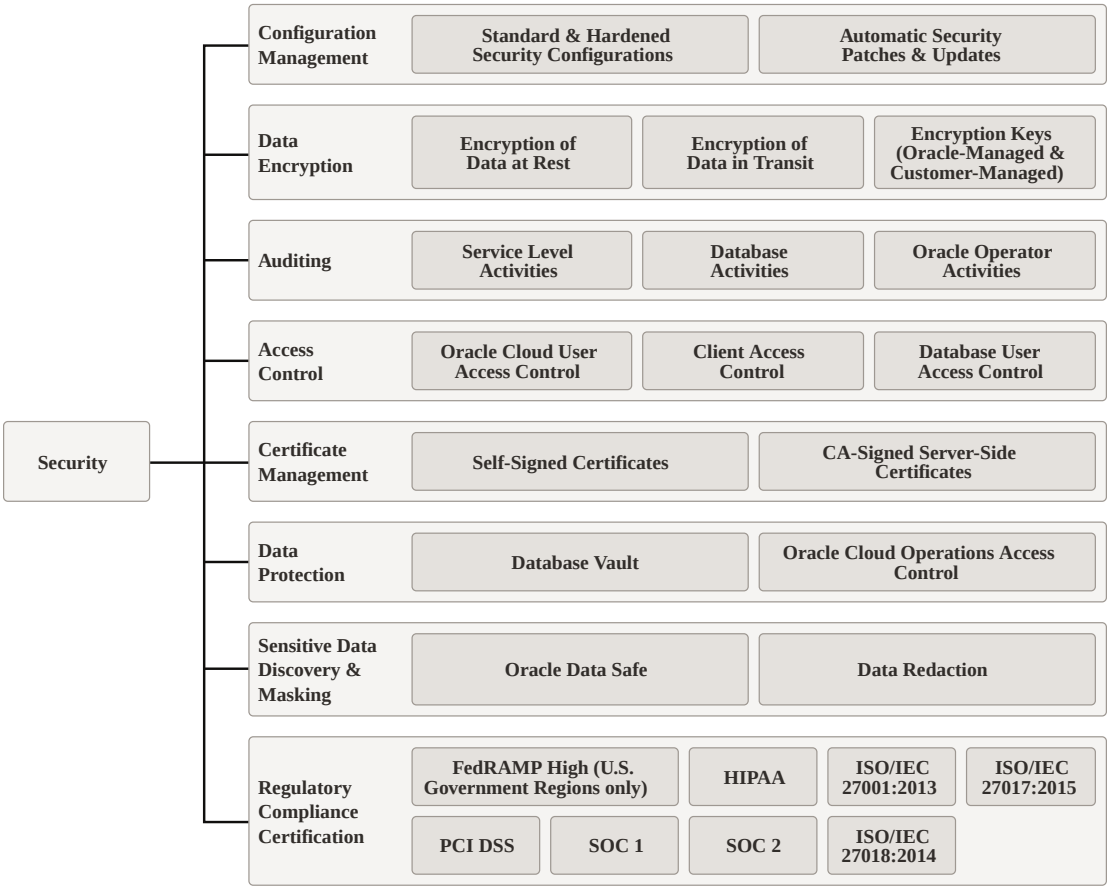
Note that throughout this section the term "you" is broadly used to mean any administrator in your organization who has the responsibility for performing certain tasks. In some cases, that's the fleet administrator, in others it's the database administrator.

Along with the standard security features of the Oracle Database such as privilege analysis, network encryption, centrally managed users, secure application roles, transparent sensitive data protection, and others, dedicated Autonomous 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 dedicated Autonomous 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 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 database fleet. Refer to Configuration Management in Autonomous 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 Database for more details.

## Data Encryption

Autonomous 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 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 Database for more details about data encryption and master encryption keys.

## Auditing

Oracle Autonomous 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 Database for more details.

## Access Control

When configuring the dedicated Exadata infrastructure feature, 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 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.

Securing access to your data and the databases that contain it comprises several different kinds of access control. Refer to Access Control Within Dedicated Autonomous Database for more details.

## Certificate Management

When a client attempts to connect to an Autonomous Database through a TCPS (Secure TCP) database connection service, Oracle Autonomous Database on Dedicated Exadata Infrastructure uses standard TLS 1.2 certificate-based authentication to authenticate the connection. 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 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 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 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 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. Refer to Manage Security Certificates for an Autonomous Exadata VM Cluster Resource for step-by-step instructions.

**Certificate management**
You can connect to Autonomous Databases associated with a CA-signed server-side with or without an Autonomous 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 Database wallets related to this Autonomous Exadata VM Cluster will expire. |
| sslcertificaterotation.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 Database on Dedicated Exadata Infrastructure for a comprehensive list of Autonomous 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 Database lets you implement **Privileged Access Management (PAM)** using:

- **Oracle Database Vault** comes preconfigured and ready to use in Autonomous 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 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 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 Database Vault Protects Privileged User Accounts in *Oracle Database Administrator's Guide* for more details.

- **Separation of duties for application data access:** Oracle Database Vault separation of duty controls can be customized and organizations with limited resources can assign multiple Oracle 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 Database Vault Addresses Database Consolidation Concerns in *Oracle Database Administrator's Guide* for more details.

Before using Database Vault, review What to Expect After You Enable Oracle Database Vault in *Oracle Database Administrator's Guide* to understand the impact of configuring and enabling Database Vault.

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

For information about how to configure and enable Database Vault in your autonomous database, see Use Oracle Database Vault to Manage Database User Privileges.

> **Tip:**
>
> To try out the process of setting up Database Vault, run *Lab 21: Protect Data With Database Vault* in Oracle Autonomous 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 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 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 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 Database. For more information about registering your database, see Register or Deregister a Dedicated 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 *PL/SQL Packages and Types Reference*.

## Regulatory Compliance Certification

Autonomous 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.

# Configuration Management in Autonomous Database on Dedicated Exadata Infrastructure

Built on Oracle Cloud Infrastructure (OCI), Autonomous 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 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 Database for more details.

## Autonomous Virtual Machine Hardening

Autonomous 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 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.

## 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 Database on Dedicated Exadata Infrastructure

Autonomous 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 database has its own encryption key, and its backups have their own different encryption key.

By default, Oracle Autonomous Database creates and manages all the master encryption keys used to protect your data, storing them in a secure PKCS 12 keystore on the same Exadata systems where the databases reside. If your company security policies require, Oracle Autonomous Database 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 Database 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 Database using Oracle Net Services (also known as SQL*Net) and predefined database connection services. Oracle Autonomous Database 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 (Transport Layer Security) protocol for connections and so use symmetric-key data encryption.

  When you create an autonomous 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 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 available by clicking **DB Connection** on the database's **Autonomous 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.

# Auditing Capabilities in Autonomous Database

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

## Auditing Service Level Activities

All actions Oracle Cloud users perform on the resources that make up your deployment of Oracle Autonomous Database 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 Database 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 Database operations that generate events, see Events for Autonomous 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 Database Dedicated for Fleet Administrators.

## Auditing Database Activities

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

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

- Audit records (including `SYS` audit records) from unified audit policies and `AUDIT` settings
- Fine-grained audit records from the `DBMS_FGA` PL/SQL package
- Oracle Database Real Application Security audit records
- Oracle Recovery Manager audit records
- Oracle Database Vault audit records
- Oracle Label Security audit records

- Oracle Data Mining records
- Oracle Data Pump
- Oracle SQL*Loader Direct Load

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 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 Security Guide*.

Additionally, if your register your autonomous 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 Database with Data Safe.

## Auditing Autonomous VM Activities

Collection agents running on the control plane servers of Autonomous 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 Database 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 Database Dedicated for Fleet Administrators Workshop.

Autonomous 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 Database, and Autonomous Exadata VM Cluster (client virtual machines deployed on Oracle Autonomous 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 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 Database. 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

When configuring the dedicated Exadata infrastructure feature, 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 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.

Securing access to your data and the databases that contain it comprises several different kinds of access control. Refer to Access Control Within Dedicated Autonomous Database for more details.

## 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 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 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 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 Database, refer to IAM Policies for Autonomous 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 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 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 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 Database by controlling network access control and client connections.

### Network Access Control

You define network access control to autonomous databases when you set up and configure you dedicated deployment of Oracle Autonomous Database. 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 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 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.

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. This allows you to formulate a fine grained access control policy by limiting your Autonomous 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 Database for details.

Note the following about using an ACL with Autonomous Database:

- The Autonomous Database Service Console is not subject to ACL rules.
- As of the current release, Oracle Application Express (APEX), RESTful services, and Database Actions are not subject to ACLs. Choosing to enable an ACL disables these features automatically.
- Performance Hub is not subject to ACL rules.
- While creating an Autonomous 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 Database operations are not allowed.

**Client Connection Control**

Oracle Autonomous Database implements client connection control with standard TLS 1.2 certificate-based authentication to authenticate a client connection.

By default, Autonomous 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 Database.

## Database User Access Control

Oracle Autonomous Database configures the databases you create to use the standard user management feature of Oracle 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 Database Concepts*. For detailed information and guidance, see Managing Security for Oracle Database Users in *Oracle Database Security Guide*.

If your database user access strategy demands more controls than are provided by standard user management, you can configure your autonomous database 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 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.<br>Refer to Data Protection in *Security Features of Autonomous Database* for more details. |
| Oracle Cloud Infrastructure Identity and Access Management (IAM) | You can configure Autonomous Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous Database with IAM credentials. Refer to Use Identity and Access Management (IAM) Authentication with Autonomous Database for using this option with your database. |
| Azure OAuth2 Access Tokens | You can centrally manage Oracle Autonomous Database 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 Database instance. Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to get an Azure AD `OAuth2` access token to send to the database.<br>For more information about integrating Microsoft Azure Active Directory with your databases, see Authenticate and Authorize Microsoft Azure Active Directory Users for Autonomous Database. |
| Microsoft Active Directory (CMU-AD) | If you use Microsoft Active Directory as a user repository, you can configure your Autonomous 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.<br>For more information about integrating Microsoft Active Directory with your databases, see Microsoft Active Directory with Autonomous Database. |

| Tool | Description |
| --- | --- |
| 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.<br>Autonomous Database support for Kerberos provides the benefits of single sign-on and centralized authentication of Oracle users. For more information, see Authenticate Autonomous 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.<br>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 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.<br>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.<br>For more information about Oracle RAS, see Introducing Oracle Database Real Application Security in *Oracle Database Real Application Security Administrator's and Developer's Guide*.<br>Oracle Autonomous Database 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 databases.<br>For more information about Virtual Private Database, see Using Oracle Virtual Private Database to Control Data Access in *Oracle Database 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 Database on Dedicated Exadata Infrastructure is designed to isolate and protect customer services and database data from unauthorized access. Autonomous 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 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 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 Databases. Oracle Cloud operations and support staff do not have access to the data in your Autonomous Databases. When you create an Autonomous Container Database, Autonomous 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 Databases created in the container database.

  You can confirm that Operations Control is active by entering this SQL statement in an Autonomous 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 Database.

# IAM Policies for Autonomous Database on Dedicated Exadata Infrastructure

This article lists the IAM policies required for managing the infrastructure resources of Autonomous Database on dedicated Exadata infrastructure.

Oracle Autonomous 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 Database

This topic covers details for writing policies to control access to Autonomous Database resources.

A policy defines what kind of access a group of users has to a specific resource in an individual compartment. For more information, see Getting Started with Policies.

> **Tip:**
>
> For a sample policy, see Let database and fleet admins manage Autonomous 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 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)*

`autonomousContainerDatabaseDataguardAssociations`

`AutonomousDatabaseDataguardAssociation`

`autonomous-virtual-machine`

> **Tip:**
>
> The `cloud-exadata-infrastructures` and `exadata-infrastructures` resource-types needed to provision Autonomous 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 Bare Metal and Virtual Machine DB Systems.

### Supported Variables

General variables are supported. See General Variables for All Requests for more information.

Additionally, you can use the `target.workloadType` variable, as shown in the following table:

| target.workloadType value | Description |
|---|---|
| `OLTP` | Online Transaction Processing, used for Autonomous Databases with Autonomous Transaction Processing workload. |
| `DW` | Data Warehouse, used for Autonomous 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 Database workload types.

#### autonomous-databases

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| inspect | `AUTONOMOUS_DATABASE_INSPECT` | `GetAutonomousDatabase`, `ListAutonomousDatabases` | *none* |
| read | *`INSPECT`* + `AUTONOMOUS_DATABASE_CONTENT_READ` | *no extra* | `CreateAutonomousDatabaseBackup` (also needs `manage autonomous-backups`) |

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| use | *READ* +<br>AUTONOMOUS_DATABASE_CONTENT_WRITE<br>AUTONOMOUS_DATABASE_UPDATE | UpdateAutonomousDatabase | RestoreAutonomousDatabase (also needs read autonomous-backups)<br>ChangeAutonomousDatabaseCompartment (also needs read autonomous-backups) |
| manage | *USE* +<br>AUTONOMOUS_DATABASE_CREATE<br>AUTONOMOUS_DATABASE_DELETE | CreateAutonomousDatabase | *none* |

### autonomous-backups

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| inspect | AUTONOMOUS_DB_BACKUP_INSPECT | ListAutonomousDatabaseBackups, GetAutonomousDatabaseBackup | *none* |
| read | *INSPECT* +<br>AUTONOMOUS_DB_BACKUP_CONTENT_READ | *no extra* | RestoreAutonomousDatabase (also needs use autonomous-databases)<br>ChangeAutonomousDatabaseCompartment (also needs use autonomous-databases) |
| use | *READ* +<br>*no extra* | *no extra* | *none* |
| manage | *USE* +<br>AUTONOMOUS_DB_BACKUP_CREATE<br>AUTONOMOUS_DB_BACKUP_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 | *none* |

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| read | *INSPECT* + *no extra* | *no extra* | *none* |
| use | *READ* + `AUTONOMOUS_CONTAINER_DATABASE_UPDATE` | `UpdateAutonomousContainerDatabase` `ChangeAutonomousContainerDatabaseCompartment` | `CreateAutonomousDatabase` (also needs `manage autonomous-databases`) |
| manage | *USE* + `AUTONOMOUS_CONTAINER_DATABASE_CREATE` `AUTONOMOUS_CONTAINER_DATABASE_DELETE` | *no extra* | `CreateAutonomousContainerDatabase`, `TerminateAutonomousContainerDatabase` (both also need `use cloud-autonomous-vmclusters`, `use cloud-exadata-infrastructures`) |

**cloud-autonomous-vmclusters**

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| inspect | `CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT` | `ListCloudAutonomousVmClusters` `GetCloudAutonomousVmCluster` | *none* |
| read | *INSPECT* + *no extra* | *no extra* | *none* |
| use | *READ* + `CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE` | `UpdateCloudAutonomousVmCluster` `ChangeCloudAutonomousVmClusterCompartment` | *CreateAutonomousDatabase* (also needs `manage autonomous-databases`) `CreateAutonomousContainerDatabase` (also needs `manage autonomous-container-databases`) |
| manage | *USE* + `CLOUD_AUTONOMOUS_VM_CLUSTER_CREATE` `CLOUD_AUTONOMOUS_VM_CLUSTER_DELETE` | *no extra* | `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<br>GetAutonomousVmCluster | ChangeAutonomousVmClusterCompartment |
| read | *INSPECT* +<br>*no extra* | *no extra* | *none* |
| use | *READ* +<br>AUTONOMOUS_VM_CLUSTER_UPDATE | ChangeAutonomousVmClusterCompartment | UpdateAutonomousVmCluster<br>CreateAutonomousContainerDatabase<br>TerminateAutonomousContainerDatabase |
| manage | *USE* +<br>AUTONOMOUS_VM_CLUSTER_CREATE +<br>AUTONOMOUS_VM_CLUSTER_DELETE | DeleteAutonomousVmCluster | CreateAutonomousVmCluster |

### autonomousContainerDatabaseDataguardAssociations

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| inspect | AUTONOMOUS_VM_CLUSTER_INSPECT<br>AUTONOMOUS_CONTAINER_DATABASE_INSPECT | GetAutonomousContainerDatabase<br>ListAutonomousContainerDatabaseDataguardAssociations<br>GetAutonomousContainerDatabaseDataguardAssociation | CreateAutonomousContainerDatabase<br>FailoverAutonomousContainerDatabaseDataguardAssociation<br>SwitchoverAutonomousContainerDatabaseDataguardAssociation<br>ReinstateAutonomousContainerDatabaseDataguardAssociation |
| read | *no extra* | *no extra* | *no extra* |

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| use | *READ* +<br>AUTONOMOUS_VM_CLUSTER_UPDATE +<br>AUTONOMOUS_CONTAINER_DATABASE_UPDATE | *none* | CreateAutonomousContainerDatabase<br>deleteAutonomouContainerDatabase<br>FailoverAutonomousContainerDatabaseDataguardAssociation<br>SwitchoverAutonomousContainerDatabaseDataguardAssociation<br>ReinstateAutonomousContainerDatabaseDataguardAssociation |
| manage | *USE* +<br>AUTONOMOUS_CONTAINER_DATABASE_CREATE +<br>AUTONOMOUS_CONTAINER_DATABASE_DELETE | *none* | CreateAutonomousContainerDatabase<br>deleteAutonomouContainerDatabase |

### AutonomousDatabaseDataguardAssociation

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| inspect | AUTONOMOUS_DATABASE_INSPECT | GetAutonomousDatabase | *none* |
| read | *no extra* | *no extra* | *no extra* |
| use | *READ* +<br>*no extra* | *no extra* | *no extra* |
| manage | *USE* +<br>*no extra* | *no extra* | *no extra* |

### autonomous-virtual-machine

| Verbs | Permissions | APIs Fully Covered | APIs Partially Covered |
|---|---|---|---|
| inspect | AUTONOMOUS_VIRTUAL_MACHINE_INSPECT | GetAutonomousVirtualMachine<br>ListAutonomousVirtualMachines | *none* |

### Permissions Required for Each API Operation

Autonomous Container Database (ACD) and Autonomous Database (ADB) are common resources between Oracle Public Cloud 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.

For information about permissions, see Permissions.

The following table lists the API operations for Autonomous Database resources in a logical order, grouped by resource type.

### Autonomous Database API Operations

| API Operation | Permissions Required to Use the Operation | Notes |
|---|---|---|
| `ListCloudAutonomousVmClusters` | CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT | APPLIES TO: ● Oracle Public Cloud only |
| `ListAutonomousVmClusters` | AUTONOMOUS_VM_CLUSTER_INSPECT | APPLIES TO: ● Exadata Cloud@Customer only |
| `GetCloudAutonomousVmCluster` | CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT | APPLIES TO: ● Oracle Public Cloud only |
| `GetAutonomousVmCluster` | AUTONOMOUS_VM_CLUSTER_INSPECT | APPLIES TO: ● Exadata Cloud@Customer only |
| `CreateCloudAutonomousVmCluster` | CLOUD_AUTONOMOUS_VM_CLUSTER_CREATE and CLOUD_EXADATA_INFRASTRUCTURE_UPDATE | APPLIES TO: ● Oracle Public Cloud only |
| `CreateAutonomousVmCluster` | AUTONOMOUS_VM_CLUSTER_CREATE and EXADATA_INFRASTRUCTURE_INSPECT and EXADATA_INFRASTRUCTURE_UPDATE | APPLIES TO: ● Exadata Cloud@Customer only |
| `UpdateCloudAutonomousVmCluster` | CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE and CLOUD_EXADATA_INFRASTRUCTURE_UPDATE | APPLIES TO: ● Oracle Public Cloud only |

| API Operation | Permissions Required to Use the Operation | Notes |
|---|---|---|
| `UpdateAutonomousVmCluster` | AUTONOMOUS_VM_CLUSTER_UPDATE and EXADATA_INFRASTRUCTURE_INSPECT and EXADATA_INFRASTRUCTURE_UPDATE | APPLIES TO: ● Exadata Cloud@Customer only |
| `ChangeCloudAutonomousVmClusterCompartment` | CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE | APPLIES TO: ● Oracle Public Cloud only |
| `ChangeAutonomousVmClusterCompartment` | AUTONOMOUS_VM_CLUSTER_INSPECT and AUTONOMOUS_VM_CLUSTER_UPDATE | APPLIES TO: ● Exadata Cloud@Customer only |
| `RotateCloudAutonomousVmClusterOrdsCerts` | CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE | APPLIES TO: ● Oracle Public Cloud only |
| `RotateCloudAutonomousVmClusterSslCerts` | CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE | APPLIES TO: ● Oracle Public Cloud only |
| `DeleteCloudAutonomousVmCluster` | CLOUD_AUTONOMOUS_VM_CLUSTER_DELETE | APPLIES TO: ● Oracle Public Cloud only |
| `DeleteAutonomousVmCluster` | AUTONOMOUS_VM_CLUSTER_DELETE | APPLIES TO: ● Exadata Cloud@Customer only |
| `ListAutonomousContainerDatabases` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT | |
| `GetAutonomousContainerDatabase` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT | |
| `CreateAutonomousContainerDatabase` | **On Oracle Public Cloud**<br>CLOUD_EXADATA_INFRASTRUCTURE_UPDATE and CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_CREATE<br>**On Exadata Cloud@Customer**<br>EXADATA_INFRASTRUCTURE_UPDATE and AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_CREATE | |

| API Operation | Permissions Required to Use the Operation | Notes |
|---|---|---|
| `TerminateAutonomousContainerDatabase` | **On Oracle Public Cloud**<br>CLOUD_EXADATA_INFRASTRUCTURE_UPDATE and CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_DELETE<br>**On Exadata Cloud@Customer**<br>EXADATA_INFRASTRUCTURE_UPDATE and AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_DELETE | |
| `UpdateAutonomousContainerDatabase` | AUTONOMOUS_CONTAINER_DATABASE_UPDATE | |
| `ChangeAutonomousContainerDatabaseCompartment` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE | |
| `RotateAutonomousContainerDatabaseEncryptionKey` | AUTONOMOUS_CONTAINER_DATABASE_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_INSPECT | APPLIES TO: Exadata Cloud@Customer only |
| `GetAutonomousDatabase` | AUTONOMOUS_DATABASE_INSPECT | |
| `ListAutonomousDatabases` | AUTONOMOUS_DATABASE_INSPECT | |
| `CreateAutonomousDatabase` | AUTONOMOUS_DATABASE_CREATE | |
| `UpdateAutonomousDatabase` | AUTONOMOUS_DATABASE_UPDATE | |
| `ChangeAutonomousDatabaseCompartment` | AUTONOMOUS_DATABASE_UPDATE and AUTONOMOUS_DB_BACKUP_INSPECT and AUTONOMOUS_DB_BACKUP_CONTENT_READ and AUTONOMOUS_DATABASE_CONTENT_WRITE | |
| `DeleteAutonomousDatabase` | AUTONOMOUS_DATABASE_DELETE | |
| `StartAutonomousDatabase` | AUTONOMOUS_DATABASE_UPDATE | |
| `StopAutonomousDatabase` | AUTONOMOUS_DATABASE_UPDATE | |
| `RestartAutonomousDatabase` | AUTONOMOUS_DATABASE_UPDATE | |

| API Operation | Permissions Required to Use the Operation | Notes |
|---|---|---|
| `RestoreAutonomousDatabase` | AUTONOMOUS_DB_BACKUP_CONTENT_READ and AUTONOMOUS_DATABASE_CONTENT_WRITE | |
| `RotateAutonomousDatabaseEncryptionKey` | AUTONOMOUS_DATABASE_UPDATE | |
| `CreateAutonomousDatabaseBackup` | AUTONOMOUS_DB_BACKUP_CREATE and AUTONOMOUS_DATABASE_CONTENT_READ | |
| `ListAutonomousDatabaseBackups` | AUTONOMOUS_DB_BACKUP_INSPECT | |
| `GetAutonomousDatabaseBackup` | AUTONOMOUS_DB_BACKUP_INSPECT | |
| `ListAutonomousContainerDatabaseDataguardAssociations` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT | |
| `GetAutonomousContainerDatabaseDataguardAssociation` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT | |
| `FailoverAutonomousContainerDatabaseDataguardAssociation` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE | |
| `SwitchoverAutonomousContainerDatabaseDataguardAssociation` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE | |
| `ReinstateAutonomousContainerDatabaseDataguardAssociation` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE | |
| `UpdateAutonomousContainerDatabaseDataguardAssociation` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE | |
| `ListAutonomousDatabaseDataguardAssociations` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT | |
| `GetAutonomousDatabaseDataguardAssociation` | AUTONOMOUS_CONTAINER_DATABASE_INSPECT | |

### 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 | Required IAM Policies on Exadata Cloud@Customer |
|---|---|---|
| Create an Exadata Infrastructure resource | `manage cloud-exadata-infrastructures`<br>`use vnic`<br>`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`<br>`use vnic`<br>`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 | Required IAM Policies on Exadata Cloud@Customer |
|---|---|---|
| Create an Autonomous Exadata VM Cluster | `manage cloud-autonomous-vmclusters`<br>`use cloud-exadata-infrastructures` | `manage autonomous-vmclusters`<br>`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`<br>`inspect exadata-infrastructures` |
| 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.

| Operation | Required IAM Policies |
|---|---|
| Create an Autonomous Container Database | `manage autonomous-container-databases`<br>`use cloud-exadata-infrastructures` if creating the Autonomous Container Database on Oracle Public Cloud.<br>`use cloud-autonomous-vmclusters` if creating the Autonomous Container Database on Oracle Public Cloud.<br>`use autonomous-vmclusters` if creating the Autonomous Container Database on Exadata Cloud@Customer.<br>`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` |

| Operation | Required IAM Policies |
| --- | --- |
| Move an Autonomous Container Database to another compartment | `use autonomous-container-databases` |
| Rotate an Autonomous Container Database encryption key | APPLIES TO: Exadata Cloud@Customer only<br>`use autonomous-container-databases`<br>`inspect autonomous-container-databases` |
| Terminate an Autonomous Container Database | `manage autonomous-container-databases`<br>`use cloud-exadata-infrastructures` if creating the Autonomous Container Database on Oracle Public Cloud.<br>`use cloud-autonomous-vmclusters` if creating the Autonomous Container Database on Oracle Public Cloud.<br>`use autonomous-vmclusters` if creating the Autonomous Container Database on Exadata Cloud@Customer. |

## Policies to Manage Autonomous Databases

The following table lists the IAM policies required for a cloud user to perform management operations on Autonomous Databases.

| Operation | Required IAM Policies |
| --- | --- |
| Create an Autonomous Database | `manage autonomous-databases`<br>`read autonomous-container-databases` |
| View a list of Autonomous Databases | `inspect autonomous-databases` |
| View details of an Autonomous Database | `inspect autonomous-databases` |
| Set the password of an Autonomous Database's ADMIN user | `use autonomous-databases` |
| Scale the CPU core count or storage of an Autonomous Database | `use autonomous-databases` |
| Enable or disable auto scaling for an Autonomous Database | `use autonomous-databases` |
| Move an Autonomous Database to another compartment | `use autonomous-databases` in the Autonomous Database's current compartment and in the compartment you are moving it to<br>`read autonomous-backups` |
| Stop or start an Autonomous Database | `use autonomous-databases` |
| Restart an Autonomous Database | `use autonomous-databases` |
| Back up an Autonomous Database manually | `read autonomous-databases`<br>`manage autonomous-backups` |
| Restore an Autonomous Database | `use autonomous-databases`<br>`read autonomous-backups` |
| Clone an Autonomous Database | `manage autonomous-databases`<br>`read autonomous-container-databases` |

| Operation | Required IAM Policies |
| --- | --- |
| Terminate an Autonomous Database | `manage autonomous-databases` |

# Master Encryption Keys in Autonomous Database on Dedicated Exadata Infrastructure

By default, Autonomous 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 Database can instead use keys you create and manage.

For Oracle Autonomous Database on Oracle Cloud, you use the Oracle Cloud Infrastructure Vault service to create and manage keys, and for Oracle Autonomous Database on Exadata Cloud@Customer you use Oracle Key Vault.

> **Caution:**
>
> Because customer-managed keys are stored in a separate key vault external to the database host, any configuration change or interruption that makes the key vault inaccessible to a database using its keys renders that database's data inaccessible.

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.

## Prepare to 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 Database; specifically:

1. Use the Vault service to create a vault containing at least one valid master encryption key.
2. 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 Database resources reside.
3. Use the IAM service to create a dynamic group identifying your Autonomous 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 Database Dedicated for Security Administrators.

### Before You Begin: Compartment Hierarchy Best Practice

Oracle recommends that you create a compartment hierarchy for your Oracle Autonomous 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 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 Oracle Autonomous 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.

### Task 1. Create a Vault and a Master Encryption Key

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 Before You Begin: Compartment Hierarchy Best Practice.

After creating the vault, 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

Oracle strongly recommends that you create a separate master encryption key for each of your Autonomous Container Databases. Doing so makes management of key rotation over time much simpler.

**Task 2. Create a Service Gateway, a Route Rule and an Egress Security Rule**

Create a service gateway in the VCN (Virtual Cloud Network) where your Autonomous 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 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.

**Task 3. Create a Dynamic Group and a Policy Statement**

To grant your Autonomous 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 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 Exadata Infrastructure 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_DELETE',
request.permission!='KEY_MOVE',
request.permission!='KEY_IMPORT',
request.permission!='KEY_BACKUP'
}
```

If you are using a 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.

## Prepare to Use Customer-Managed Keys in Oracle Key Vault

Oracle Key Vault 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 Oracle Key Vault deployment with Oracle Autonomous Database on Exadata Cloud@Customer to create and manage your own the master keys.

Before you can use customer-managed keys stored in Oracle Key Vault, you must perform a number of preparatory configuration tasks as described in Setting Up Your Exadata Cloud@Customer to Work With Oracle Key Vault in *Exadata Cloud@Customer Configuration and Administration Guide*.

## Use Customer-Managed Keys

Learn how to use customer-managed keys in Oracle Autonomous Database.

After performing the necessary configuration steps to enable using customer-managed keys in Oracle Autonomous Database instead of Oracle-managed keys, the technique to actually use them is quite simple:

1. Create an Autonomous Container Database that specifies the use of customer-managed keys instead of Oracle-managed keys.
2. Create Autonomous Databases in the Autonomous Container Database.

The Autonomous Databases you create automatically use customer-managed keys because the Autonomous Container Database in which they are created is configured to use customer-managed keys. Thus, those users who create and manage Autonomous Databases do not have to worry about configuring their databases to use customer-managed keys.

See Create an Autonomous Container Database to create an Autonomous Container Database that uses customer-managed keys.

# Features for Experienced Database Users

## Oracle Database Features in Autonomous Database on Dedicated Exadata Infrastructure

This article provides information on using Oracle Database features and options in Autonomous Database on Dedicated Exadata Infrastructure.

For equivalent information in Autonomous Database Serverless deployments, see Autonomous Database for Experienced Database Users.

Autonomous Database configures and optimizes your database for you. You do not need to perform administration operations for configuring the database. SQL commands used 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 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 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 Transaction Processing workloads, compression is *not* enabled by default. For Autonomous Data Warehouse 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 Transaction Processing workloads. However, for the Autonomous Data Warehouse 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 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 Database Features

Autonomous Database includes the latest Oracle Database features.

Autonomous Database includes features that:

- Automate index management tasks, such as creating, rebuilding, and dropping indexes based on changes in the application workload. See Managing Auto Indexes in *Oracle Database Administrator's Guide* for more information.
- Gather real-time statistics automatically while a conventional DML workload is running. Because statistics can go stale between `DBMS_STATS` jobs, online statistics

gathering for conventional DML helps the optimizer generate more optimal plans. Online statistics aim to reduce the possibility of the optimizer being misled by stale statistics. You can manage and access statistics for conventional DML through PL/SQL packages, data dictionary views, and hints. See Real-Time Statistics in *Oracle Database SQL Tuning Guide* for more information.

- Gather statistics automatically on a more frequent basis. High-frequency automatic optimizer statistics collection complements the standard statistics collection job. By default, the collection occurs every 15 minutes, meaning that statistics have less time in which to be stale. See Configuring High-Frequency Automatic Optimizer Statistics Collection in *Oracle Database SQL Tuning Guide* for more information.
- Quarantine execution plans for SQL statements that are terminated by the Resource Manager for consuming excessive system resources in an Oracle database. You can configure quarantine settings for a SQL statement by specifying limits on its resource consumption using procedures in the `DBMS_SQLQ` package. When the SQL statement crosses any of these resource consumption limits, it is terminated and the execution plan is quarantined. In this way, the database prevents high-resource SQL statements from executing repeatedly. See Quarantine for Execution Plans for SQL Statements Consuming Excessive System Resources in *Oracle Database Administrator's Guide* for more information.
- Optimize high-frequency single-row inserts for applications, such as Internet of Things (IoT) applications. See Enabling High Performance Data Streaming With the Memoptimized Rowstore in *Oracle Database Performance Tuning Guide*.

## The ADMIN User and the SYS User

In Oracle Autonomous Database 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 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 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 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 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
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
TIME_ZONE
```

# SYSDATE_AT_DBTIMEZONE

SYSDATE_AT_DBTIMEZONE enables special handling in a session for the date and time value returned in calls to SYSDATE and SYSTIMESTAMP.

Depending on the value of SYSDATE_AT_DBTIMEZONE, you see either the date and time based on the default Autonomous Database time zone, Coordinated Universal Time (UTC), or based on the time zone that you set in your database.

| Property | Description |
|---|---|
| Parameter type | Boolean |
| Default Value | FALSE |
| Modifiable | ALTER SESSION, ALTER SYSTEM |
| Range of values | TRUE \| FALSE |

### Default Autonomous Database Time Zone

The default Autonomous Database time zone is Coordinated Universal Time (UTC) and by default calls to SYSDATE and SYSTIMESTAMP return the date and time in UTC.

In order to change database time zone, you can run the following statement. This example sets the database time zone to UTC-5.

```
ALTER DATABASE SET TIME_ZONE='-05:00';
```

> **Note:**
>
> You must restart the Autonomous Database instance for the change to take effect.

After you set the database time zone, by default SYSDATE and SYSTIMESTAMP continue to return date and time in UTC (SYSDATE_AT_DBTIMEZONE is FALSE by default). If you set SYSDATE_AT_DBTIMEZONE to TRUE in a session, SYSDATE and SYSTIMESTAMP return the database time zone.

See Setting the Database Time Zone for more information on using the SET TIME_ZONE clause with ALTER DATABASE.

### Using SYSDATE_AT_DBTIMEZONE in a Session

When SYSDATE_AT_DBTIMEZONE is FALSE in a session, calls to SYSDATE and SYSTIMESTAMP return values based on the default Autonomous Database time zone, Coordinated Universal Time (UTC).

When SYSDATE_AT_DBTIMEZONE is TRUE in a session, calls to SYSDATE or SYSTIMESTAMP return the date and time based on the database time zone.

> **Note:**
>
> Setting `SYSDATE_AT_DBTIMEZONE` to `TRUE` only affects the use of `SYSDATE` and `SYSTIMESTAMP` as operators in application SQL (for example, in queries, DML, and CTAS operations). When using this parameter, it is recommended that your client/session timezone matches your database timezone.

### Example

The following example returns dates and times for two different time zones, based on the `SYSDATE_AT_DBTIMEZONE` parameter value:

```
SQL> SELECT DBTIMEZONE FROM DUAL;

DBTIMEZONE
_____________
-05:00

SQL> ALTER SESSION SET SYSDATE_AT_DBTIMEZONE=FALSE;

Session altered.

SQL> SELECT SYSTIMESTAMP FROM DUAL;

SYSTIMESTAMP
_____________________________________
27-JAN-22 06.59.45.708082000 PM GMT

SQL> ALTER SESSION SET SYSDATE_AT_DBTIMEZONE=TRUE;

Session altered.

SQL> SELECT SYSTIMESTAMP FROM DUAL;

SYSTIMESTAMP
_____________________________________
27-JAN-22 02.14.47.578946000 PM -05:00
```

> **Note:**
>
> When a `SYSDATE` or `SYSTIMESTAMP` query is executed in SQL Worksheet of Database Actions, the time and date value that is returned is in UTC (when `SYSDATE_AT_DBTIMEZONE` parameter is set to `TRUE` or `FALSE`). To obtain the database time zone when working in Database Actions, use `TO_CHAR()` as follows:
>
> ```
> SQL> SELECT TO_CHAR(SYSTIMESTAMP,'YYYY-MM-DD"T"HH24:MI:SS
> TZH":"TZM') FROM DUAL;
>
> TO_CHAR(SYSTIMESTAMP,'YYYY-MM-DD"T"HH24:MI:SSTZH":"TZM')
> ____________________________________________________________
> 2022-01-27T14:15:00 -05:00
> ```

# Oracle Database Features with Limited Support in Autonomous Database on Dedicated Exadata Infrastructure

Autonomous 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 Database, some features have only limited support.

The following Oracle Database features, options and packs have limited support in Autonomous 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.<br>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:<br>• OML4SQL on all database versions.<br>• Oracle Data Miner.<br>For OML product documentation, refer to Oracle Machine Learning on Oracle Help Center. |
| Oracle Real Application Testing | *SQL Performance Analyzer* and *Database Replay* 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.<br>• **Database Replay:** See Database Replay for more details.<br>• **SQL Performance Analyzer (SPA):** See Testing Guide for more information on SPA.<br>**Tip:**<br>You can use the SPA features from Enterprise Manager (EM). For more details, see Monitoring and Administration Tasks in *Administrator's Guide for Oracle Autonomous Databases* |
| Oracle Spatial | See Limitations on the Use of Oracle Spatial. |
| Oracle Text | See Oracle Text. |
| Oracle XML DB | See Oracle XML DB. |

## SQL Commands

Autonomous Database on Dedicated Exadata Infrastructure allows most of the SQL commands available in Oracle Database. To ensure the security and the performance of Autonomous Database, some SQL commands are restricted.

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

The following SQL commands and PL/SQL packages are not supported in Autonomous Databases:

- `ADMINISTER KEY MANAGEMENT`
- `SYS.DBMS_SHARED_POOL`
- `CREATE_LIBRARY`

### SQL Commands with Restrictions in Autonomous Database

The following DDL commands are available in Autonomous Database with restrictions:

| SQL Command | Restrictions |
|---|---|
| `ALTER PLUGGABLE DATABASE` and `ALTER DATABASE` | Only the following clauses are allowed:<br>`DATAFILE AUTOEXTEND ON`<br>`DATAFILE AUTOEXTEND OFF`<br>`DATAFILE RESIZE`<br>`DEFAULT EDITION`<br>`SET TIME_ZONE` |
| `ALTER PROFILE` | Using `ALTER PROFILE`:<br>• There are restrictions for a user defined `PASSWORD_VERIFY_FUNCTION`. See Manage Password Complexity on Autonomous Database for more information.<br>• The optional `CONTAINER` clause is ignored if specified. |
| `ALTER SESSION` | Only the following clauses are allowed:<br>`ADVISE COMMIT, ADVISE ROLLBACK, ADVISE NOTHING`<br>`CLOSE DATABASE LINK`<br>`ENABLE COMMIT IN PROCEDURE, DISABLE COMMIT IN PROCEDURE`<br>`ENABLE PARALLEL <QUERY\|DDL\|DML>, DISABLE PARALLEL <QUERY\|DDL\|DML>, FORCE PARALLEL <QUERY\|DDL\|DML>`<br>`ENABLE RESUMABLE, DISABLE RESUMABLE`<br>`SET DEFAULT_COLLATION`<br>`SET EDITION`<br>`SET ISOLATION_LEVEL`<br>`SET ROW ARCHIVAL VISIBILITY`<br>`SET` *`init-param = value`*<br>For a list of initialization parameters you can set in Oracle Autonomous 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 `ISSES_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. |

| SQL Command | Restrictions |
|---|---|
| ALTER TABLESPACE | The use of SMALLFILE tablespaces is not supported by Autonomous Database.<br>The Autonomous 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). |
| ALTER TABLE | The following clauses are ignored:<br>allocate_extent<br>alter_iot<br>deallocate_unused<br>ilm<br>inmemory_table<br>logging<br>modify_LOB_storage<br>physical_attributes<br>shrink<br>For more information on ALTER TABLE, see *Oracle Database 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.<br>Database links can only be used for TCP connections because TCPS connections require a wallet. |
| CREATE PROFILE | Refer to Manage Password Complexity on Autonomous Database for the Password Verify Function requirements. |
| CREATE TABLE | XMLType tables using XML schema-based storage are not allowed. See Oracle XML DB.<br>The following clauses are ignored:<br>• cluster<br>• ilm_clause<br>• inmemory_table_clause<br>• LOB_storage_clause<br>• logging_clause<br>• organization external: Use of this clause has certain limitations, as described in Manually Created External Tables.<br>• 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.<br>• physical_properties<br>For more information on CREATE TABLE, see *Oracle Database SQL Language Reference*. |

| SQL Command | Restrictions |
|---|---|
| `CREATE TABLESPACE` | The use of `SMALLFILE` tablespaces is not supported by Autonomous Database.<br>The Autonomous 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:<br>• `IDENTIFIED EXTERNALLY`<br>• `IDENTIFIED BY VALUES` |
| `DROP TABLESPACE` | This command is supported only when used in the following format:<br>`DROP TABLESPACE <tablespace_name> INCLUDING CONTENTS AND DATAFILES;`<br>If you execute this command without `INCLUDING CONTENTS AND DATAFILES` in the syntax, you will get the following error:<br>`ORA-01031: insufficient privileges` |

## Data Types

Autonomous Database on Dedicated Exadata Infrastructure allows most of the data types available in Oracle Database. To ensure the security and the performance of Autonomous Database, some data types are restricted.

The following Oracle Database data types are not supported or are only partially supported in Autonomous 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 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 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 In-Memory Guide.

Autonomous Database on Dedicated Exadata Infrastructure lets you enable Database In-memory for your Autonomous 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 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 Database:

- Turning Database In-Memory on or off is supported only on Autonomous Databases with:
  - Version is **19.20 or later**.
  - **At least 4 OCPUs enabled**.
- Scaling down an Autonomous Database to less than 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 Databases.
- If you enable Database In-Memory for a primary Autonomous 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 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 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.
- Partitioned external tables and partitioned hybrid tables are not currently 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 Utilities*.

For information about LogMiner configuration and requirements, see Introduction to LogMiner in *Oracle Database Utilities*.

Autonomous 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 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 Database.
- Only one LogMiner session is allowed per Autonomous Database.
- By default, Autonomous 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"`.

## Limitations on the Use of Oracle Spatial

Autonomous Database on Dedicated Exadata Infrastructure includes Oracle Spatial, with the following limitations.

The following Oracle Spatial components require that you download applications from Oracle Cloud Marketplace:

- Oracle Spatial Studio
- Oracle Spatial Map Visualization
- Network Data Model
- Open Geospatial Consortium Web Services (WFS, WCS, CSW)

The following are not available in Autonomous Database:

| Spatial Feature | More Information |
|---|---|
| Routing Engine | Routing Engine |
| Geocoder (mid-tier) | Geocoding Address Data |

## Oracle Text

The following describes support for Oracle Text features in Autonomous Database on Dedicated Exadata Infrastructure. To ensure the security and the performance of Autonomous Database, some Oracle Text features are restricted.

| Oracle Text Feature | Supported in Autonomous Database | More Information |
|---|---|---|
| All logging, and APIs which perform logging such as `ctx_report.query_log_summary` | Not Supported | QUERY_LOG_SUMMARY |
| File and URL datastore | Not Supported | Datastore Type |
| `CREATE INDEX` with `BIG_IO` option | Not supported by default. [1] | Improved Response Time Using the BIG_IO Option of CONTEXT Index |
| `OPTIMIZE_INDEX` in rebuild mode | Not supported by default.(see Footnote 1). | OPTIMIZE_INDEX |

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

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

## Oracle XML DB

The following describes support for Oracle XML DB features in Autonomous Database on Dedicated Exadata Infrastructure. To ensure the security and the performance of Autonomous Database, some Oracle XML DB are restricted.

> **Note:**
>
> If you migrate tables containing `XMLType` columns to Autonomous 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 Database | More Information |
|---|---|---|---|
| Repository | XML DB Protocol | No | Repository Access Using Protocols |
| Repository | XML DB Resources | No | Oracle XML DB Repository Resources |
| Repository | XML DB ACLs | No | Repository Access Control |
| Storage | XML Schema Registration | No | XML Schema Registration with Oracle XML DB |
| Storage | CLOB | No | Deprecated |
| Storage | Object Relational | No | XML Schema and Object-Relational XMLType |
| Storage | Binary XML | Yes (Non schema-based only) | XMLType Storage Models |
| Index | Structured XML Index | Yes | XMLIndex Structured Component |
| Index | XQuery Full Text Index | Yes | Indexing XML Data for Full-Text Queries |
| Index | Unstructured XMLIndex | No | XMLIndex Unstructured Component |
| Packages | XML DOM package | Yes | PL/SQL DOM API for XMLType (DBMS_XMLDOM) |
| Packages | XML Parser Package | Yes | PL/SQL Parser API for XMLType (DBMS_XMLPARSER) |
| Packages | XSL Processor (DBMS_XSLPROCESSOR) | Yes | PL/SQL XSLT Processor for XMLType (DBMS_XSLPROCESSOR) |

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

## Oracle Flashback and Restore Points

The following describes support for Oracle Flashback and Restore Points in Autonomous Database on Dedicated Exadata Infrastructure. To ensure the security and the performance of Autonomous 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 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 and Restore Points Feature | Supported in Autonomous Database on Dedicated Exadata Infrastructure |
|---|---|
| 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 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 Testing Guide*.

Using the Database Replay component of Real Application Testing (RAT) in the Oracle Autonomous Database on Dedicated Exadata Infrastructure is the same as its implementation in the Oracle Database. See Part II Database Replay for the process involved in using Database Replay with your database.

There are certain limitations and recommendations to use Database Replay with Autonomous 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 Database to store the capture and replay files. Refer to Attach Network File System and Attach External File System for guidance.
- Database Replay can be used with Autonomous Database versions **19.21 and above** only.
- Replay client must be run with Patch 35341947 from **19.19 DB Home** or **19.19 Instant Client**.
- Autonomous 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 23c instance** for better compatibility.

## Database PL/SQL Packages

The following describes notes to consider before, during, and after using PL/SQL packages in Autonomous Database on Dedicated Exadata Infrastructure.

Lists Oracle Database PL/SQL packages with restrictions and notes in Autonomous Database on Dedicated Exadata Infrastructure.

### Removed PL/SQL Packages

- DBMS_DEBUG_JDWP
- DBMS_DEBUG_JDWP_CUSTOM
- UTL_TCP

### 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 SET_WALLET Procedure and UTL_HTTP Session Settings in *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 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.

## Oracle Database Features That Are Not Supported in Autonomous Database on Dedicated Exadata Infrastructure

Autonomous 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 Database, the following Oracle Database features, options, and packs are not supported in Autonomous 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 Database. See Customizing Locale Data for more information.
- `DBMS_DEBUG_JDWP` and `DBMS_DEBUG_JDWP_CUSTOM` PL/SQL packages
- `UTL_TCP` PL/SQL package
- 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 R Enterprise
- Oracle Workspace Manager
- Root container (CDB$ROOT) access
- Transportable tablespaces
- Uniform extent allocation
- XStream

## Character Set Selection for Autonomous Database

Oracle recommends choosing Unicode for its universality and compatibility with contemporary and future technologies and language requirements. Autonomous 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 Database is quite complex. See Character Set Migration and Support Note 788156.1 for more details regarding the complexity of the exercise.

Autonomous Database lets you choose a character set of your choice from a list of supported character sets while provisioning an Autonomous 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 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 Database, see step 12 in Create a Dedicated Autonomous Database.

# Use Oracle Graph with Autonomous Database on Dedicated Exadata Infrastructure

Oracle Graph enables you to create graphs from data in an Autonomous 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.

A Autonomous 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 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 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 for more details.

# Use Oracle Spatial in Autonomous Database

Spatial features are included in Autonomous Database on Dedicated Exadata Infrastructure, allowing developers and analysts to get started easily with location intelligence analytics and mapping services.

## About Oracle Spatial

Autonomous Database includes Oracle Spatial. Oracle Spatial allows developers and analysts to use spatial analysis in every application, from basic spatial search and analysis to advanced geospatial applications and Geographic Information Systems (GIS).

With the spatial features in Autonomous 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 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 and Oracle Database 19c Spatial and Graph for more information.

## Limitations on the Use of Oracle Spatial

Autonomous Database on Dedicated Exadata Infrastructure includes Oracle Spatial, with the following limitations.

The following Oracle Spatial components require that you download applications from Oracle Cloud Marketplace:

- Oracle Spatial Studio
- Oracle Spatial Map Visualization
- Network Data Model
- Open Geospatial Consortium Web Services (WFS, WCS, CSW)

The following are not available in Autonomous Database:

| Spatial Feature | More Information |
|---|---|
| Routing Engine | Routing Engine |
| Geocoder (mid-tier) | Geocoding Address Data |

# Use Advanced Queuing with Autonomous Database

Autonomous 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. For an *overview of queuing*, see What is Queuing? in *Advanced Queuing User's Guide*.

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 Leverages Oracle Database in *Advanced Queuing User's Guide*.

### Implementing Message Queuing in Autonomous 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 see *Example 8-1: Setting Up AQ Administrative Users* in List of Examples.
- **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.

> **Tip:**
>
> For a "try it out" alternative that demonstrates how to use Advanced Queuing, you can follow the instructions outlined in Building Microservices with Oracle Converged Database Workshop.

# Use JSON Documents with Autonomous Database on Dedicated Exadata Infrastructure

Autonomous 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 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 Database:

1. Create a JSON column with an `is json format oson` check constraint
2. Insert JSON data into the column, and
3. Query the JSON data

See Work with JSON Documents in Autonomous Database 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 Database to store JSON data, you can take advantage of all the features available in your Autonomous 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 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.

To use the MongoDB API with an Autonomous 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.

Perform the below listed tasks to enable Oracle Database API for MongoDB using a *customer managed ORDS* on an Autonomous Database.

1. **Install and configure ORDS using the `ords install adb` command.**

   Refer to Installing and Configuring Customer Managed ORDS on Autonomous Database in *Oracle REST Data Services Installation and Configuration Guide* for more details.

2. **Create an ORDS-enabled user.**
3. **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*.

### 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 Database.

- Go to the **Details** page of the Autonomous Database. See View Details of a Dedicated Autonomous 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 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 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 Database to learn how to use the default and customized collection metadata.

### Limitations on the Use of SODA and JSON

Autonomous 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 for more information.
- Automatic indexing is not supported for SODA query-by-example (QBE).

# 4
# Tasks

## About Connecting to Autonomous Database on Dedicated Exadata Infrastructure

Applications and tools connect to Autonomous 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 Database**

Autonomous 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 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 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 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 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 Database.

Oracle Net Services can find the location of the Autonomous Database wallet using the `WALLET_LOCATION` parameter in the `sqlnet.ora` file.When `WALLET_LOCATION` is used, Oracle Net Services automatically uses the wallet. The wallet is used transparently to the application. See Prepare for ODBC and JDBC Oracle Call Interface (OCI) Connections for information on setting `WALLET_LOCATION`.

## Predefined Database Service Names for Autonomous Databases

Autonomous 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. 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.

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

- **tpurgent_tls**, **tpurgent**, **tpurgent_ro_tls**, **tpurgent_ro**, **tpurgent_ss_tls**, and **tpurgent_ss**: For high priority, time critical transaction processing operations.
- **tp_tls**, **tp**, **tp_ro_tls**, **tp_ro**, **tp_ss_tls**, and **tp_ss**: For typical transaction processing operations.
- **high_tls**, **high**, **high_ro_tls**, **high_ro**, **high_ss_tls**, and **high_ss**: For high priority reporting and batch operations.
- **medium_tls**, **medium**, **medium_ro_tls**, **medium_ro**, **medium_ss_tls**, and **medium_ss**: For typical reporting and batch operations.
- **low_tls**, **low**, **low_ro_tls**, **low_ro**, **low_ss_tls**, and **low_ss**: For low priority reporting and batch operations.

Autonomous Transaction Processing support all the above connection services (**tpurgent**, **tp**, **high**, **medium**, and **low**). In contrast, Autonomous Data Warehouse 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 Transaction Processing, and
- The **low** services for Autonomous Data Warehouse.

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 Database.

#### Key Characteristics of the Predefined Database Services

- **Concurrent Statements:** The maximum number of SQL statements that an Autonomous Database can be processing for a consumer group before queuing the statements.
- **Degree of Parallelism:** Parallel execution is one of the Autonomous 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 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 × 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 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 Database Using a Client Application

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

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

1. Determine what connection type your application uses, (for example OCI, ODBC, JDBC Thin, and so on).
2. 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 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 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.

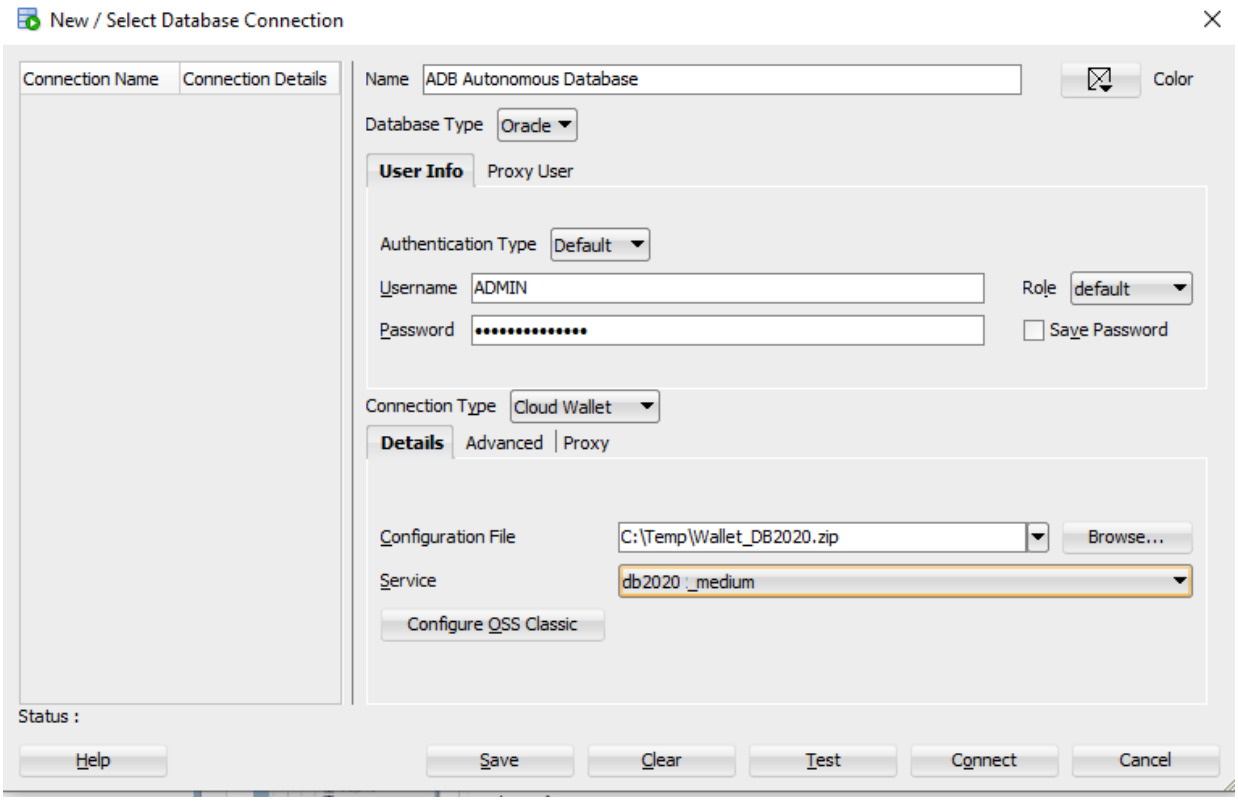
   The steps required to prepare the client computer depend on the type of connection used by the client application. In all cases, client credentials in the form of the wallet file must be downloaded to the client.

**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 Database connection, for example, Oracle SQL Developer, Oracle recommends that you use that type of connection.




# Connect to Autonomous Database Using Oracle Database Tools

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

The following topics provide step-by-step instructions for connecting to Autonomous Database using Oracle Database tools:

- **Database Actions:** Connect to Autonomous Database with Oracle Database Actions
- **Oracle SQL Developer:** Connect to Autonomous Database with Oracle SQL Developer
- **Oracle SQLcl:** Connect to Autonomous Database with Oracle SQLcl
- **SQL*Plus:** Connect to Autonomous Database with SQL*Plus

# Connect with Microsoft .NET, Visual Studio, and Visual Studio Code

Oracle Autonomous 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 Database. ODP.NET has the following driver types:

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

Oracle Developer Tools for Visual Studio provides database application design-time support in Visual Studio, including tools for managing Autonomous Databases in 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 Database.

**Set-up Instructions**

Refer to Developing .NET Applications for Oracle Autonomous Database for instructions on how to download, install, and configure these components for use with Oracle Autonomous 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 Database on Dedicated Exadata Infrastructure and .NET, try the free .NET Development with Oracle Autonomous Database Quick Start. This lab walks you through setting up a .NET web server on Oracle Cloud Infrastructure that connects to Autonomous Database. Next, the lab guides developing and deploying a simple ASP.NET Core web application that uses all these components. By the end, you will have a live, working website on the Internet.

# Connect to Autonomous 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 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 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 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 Database with Oracle Cloud Infrastructure FastConnect see FastConnect Overview.

# Migrate

## Attach External Storage to Autonomous Database

### Attach Network File Storage to Autonomous Database on Dedicated Exadata Infrastructure

You can attach your Network File Storage (NFS) share to your Autonomous Database, provided the database and the file system are in the same private network.

Supporting Network File System (NFS) allows you to do the following:

- Load highly sensitive data from other databases to Autonomous 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 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 Database, ensure that:

- The Autonomous Database version is 19.18 or later.

- 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 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 Database. You must have `WRITE` privilege on the directory object on your Autonomous Database instance to attach a file system to a directory location in the database.

   For example, the following command creates the database directory named `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 Database. To run this procedure, you must be logged in as the ADMIN user or have `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

   ```
   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;

   /
   ```

   This example attaches the network file system specified in the `file_system_name` parameter to the Autonomous Database.

   The `file_system_location` parameter specifies the location of the file system. The value you supply with `file_system_location` consists of a Fully Qualified Domain Name (FQDN) and a file path in the form: FQDN:*file_path*.

   For example:

- – FQDN: `myhost.sub000445.myvcn.oraclevcn.com`
  - – File Path: `/results`

  The `directory_name` parameter specifies the directory name in the Autonomous Database where you want to attach the file system. This is the directory you created in Step 1, or another directory you previously created.

  The `description` parameter specifies the description for the task.

Notes for using `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM`:

- Oracle Cloud Infrastructure File Storage uses NFS Version 3 to share.
- If you attach to non-Oracle Cloud Infrastructure File Storage systems, the procedure supports NFS Version 2 and NFS Version 3.
- `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` does not support NFS Version 4.

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
- 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 | Description |
|---|---|
| `FILE_SYSTEM_NAME` | File system name |
| `FILE_SYSTEM_LOCATION` | File system location |
| `DIRECTORY_NAME` | Attached directory name |
| `DIRECTORY_PATH` | Attached directory path |
| `DESCRIPTION` | The value provided for the description parameter when you run `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` |
| `CREATION_TIME` | Creation 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 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 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 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 Database on Dedicated Exadata Infrastructure

You can attach an external file storage deployed in Oracle Cloud Infrastructure or on-premises to your Autonomous Database, provided the database and the file system are in the same private network.

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 Database instance from a legacy application and use the file system to load and unload data.
- Analyze data from different sources in an Autonomous Database.
- Secure access to data in an Autonomous Database from the file systems in an on-premises data center or Private VCNs.

### External File Storage Requirements

To attach an external file storage successfully, ensure that your Autonomous Database version is 19.18 or later.

To access data in an Autonomous 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) protocol

    See 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 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 Database. You must have `WRITE` privilege on the directory object on your Autonomous Database instance to attach a file system to a directory location in the database.

   For example, the following command creates the database directory named `FSS_DIR` and creates the file system directory `fss`:

   ```
   CREATE DIRECTORY FSS_DIR AS 'fss';
   ```

   See Create a Directory for more information.

2. Run `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` to attach a file system to a directory in your Autonomous Database. To run this procedure, you must be logged in as the ADMIN user or have `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

```
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 FSS for sales
data'
);END;

/
```

   This example attaches the network file system specified in the `file_system_name` parameter to the Autonomous Database.

   The `file_system_location` parameter specifies the location of the file system. The value you supply with `file_system_location` consists of a Fully Qualified Domain Name (FQDN) and a file path in the form: FQDN:*file_path*. For example:

   - FQDN: `myhost.sub000445.myvcn.oraclevcn.com`
   - File Path: `/results`

   The `directory_name` parameter specifies the directory name in the Autonomous Database where you want to attach the file system. This is the directory you created in Step 1, or another directory you previously created.

   The `description` parameter specifies the description for the task.

Notes for using `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM`:

- Oracle Cloud Infrastructure File Storage uses NFS Version 3 to share.
- If you attach to non-Oracle Cloud Infrastructure File Storage systems, the procedure supports NFS Version 2 and NFS Version 3.
- `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` does not support NFS Version 4.

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
- 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 | Description |
|---|---|
| FILE_SYSTEM_NAME | File system name |
| FILE_SYSTEM_LOCATION | File system location |
| DIRECTORY_NAME | Attached directory name |
| DIRECTORY_PATH | Attached directory path |
| DESCRIPTION | The value provided for the description parameter when you run DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM |
| CREATION_TIME | Creation 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 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 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 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 Database

## Migrate Data into Autonomous Database on Dedicated Exadata Infrastructure

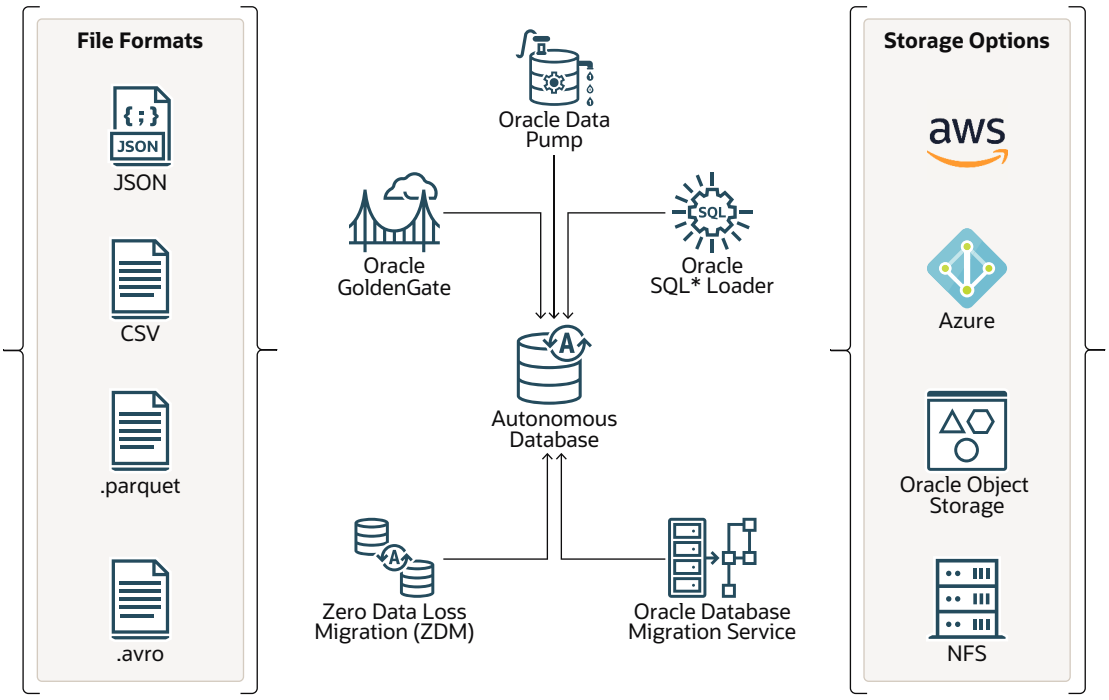
Data migration is moving data with or without its schema from one system, location, or application to another. One classic example of data migration is when you decide to adopt Autonomous Database for your organization and move the existing historical data from the current database to Autonomous Database.

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 Database.

### Data Migration Overview

Oracle provides several options to migrate your data to an Autonomous Database on Dedicated Exadata Infrastructure. Autonomous 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 Database, you can:

- Use traditional tools such as Oracle SQL*Loader and Oracle Data Pump to migrate your source databases into Autonomous 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 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 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`.

## 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 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 for more information.

## Migration Options

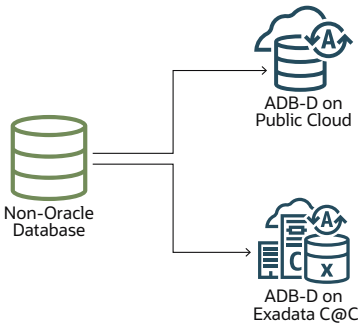
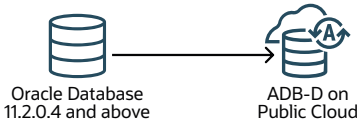
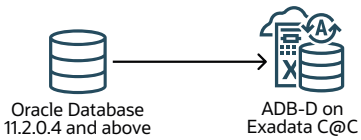
There are multiple options available to move your data into an Autonomous Database on Dedicated Infrastructure in the public cloud as well as Cloud@Customer.

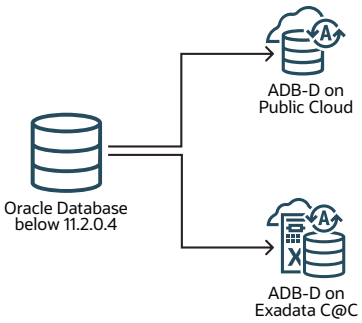
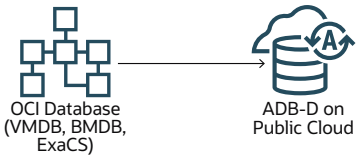
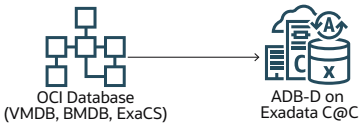
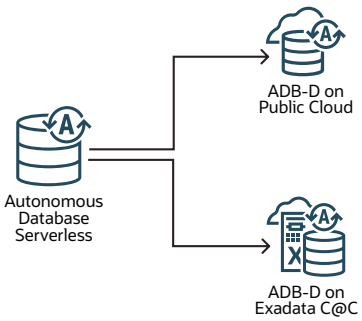
These options are listed below along with a table to pick the right option based on customer use case.

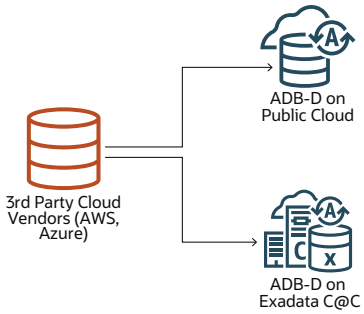
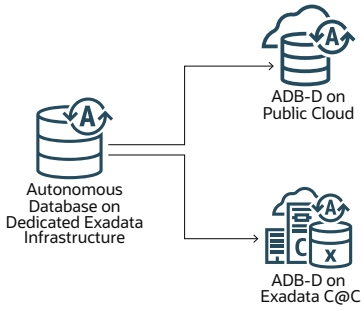
- **Oracle Database Migration Service:** A fully managed cloud service that simplifies moving established databases from on-premise, third-party, or on Oracle Cloud to Autonomous Database.
- **Oracle GoldenGate:** Oracle's data replication tool for one time data migration as well as data replication 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 Database.
- **Oracle Data Pump:** Database utility supporting high-speed bulk data and metadata movement between Oracle databases and Autonomous 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 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.<br>  <br> | • Oracle SQL*Loader<br>• Oracle GoldenGate<br>• Using NFS storage to migrate data into Autonomous Database on Exadata Cloud@Customer<br>• Using OCI File Storage to load data into Autonomous Database on Oracle Public Cloud | • Load Data Using SQL*Loader<br>• Use Oracle GoldenGate to Replicate Data with Autonomous Database<br>• Attach Network File Storage to Autonomous Database<br>• Attach External File Storage to Autonomous Database |
| Oracle Database version 11.2.0.4 and above → ADB-D on Public Cloud<br>  <br> | • Database Migration Service (DMS)<br>• Zero Downtime Migration (ZDM) - Online or offline migration using OCI Object Storage<br>• Oracle Data Pump<br>• Oracle SQL*Loader<br>• Oracle GoldenGate | • Migrate to Autonomous Database Using Oracle Database Migration Service<br>• Logical Migrations with Zero Downtime Migration<br>• Load Data Using Oracle Data Pump<br>• Load Data Using SQL*Loader<br>• Use Oracle GoldenGate to Replicate Data with Autonomous Database |
| Oracle Database version 11.2.0.4 and above → ADB-D on Exadata Cloud@Customer<br>  <br> | • Zero Downtime Migration (ZDM) - Online or offline migration using OCI Object Storage<br>• Using NFS storage to migrate data into Autonomous on Exadata Cloud@Customer<br>• Oracle Data Pump using OCI Object Storage or Customer provided NFS Storage<br>• Oracle GoldenGate | • Logical Migrations with Zero Downtime Migration<br>• Use Oracle GoldenGate to Replicate Data with Autonomous Database<br>• Attach Network File Storage to Autonomous 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<br>  | • Oracle SQL*Loader<br>• Oracle Data Pump<br>• Using NFS storage to migrate data into Autonomous on Exadata Cloud@Customer<br>• Using OCI File Storage to load data into Autonomous Database on Oracle Public Cloud | • Load Data Using SQL*Loader<br>• Load Data Using Oracle Data Pump<br>• Attach Network File Storage to Autonomous Database<br>• Attach External File Storage to Autonomous Database |
| OCI Database (VMDB, BMDB, ExaCS) → ADB-D on Public Cloud<br>  | • Database Migration Service (DMS)<br>• Zero Downtime Migration (ZDM) - Online or offline migration using OCI Object Storage<br>• Oracle Data Pump<br>• Oracle SQL*Loader | • Migrate to Autonomous Database Using Oracle Database Migration Service<br>• Logical Migrations with Zero Downtime Migration<br>• Load Data Using Oracle Data Pump<br>• Load Data Using SQL*Loader |
| OCI Database (VMDB, BMDB, ExaCS) → ADB-D on Exadata Cloud@Customer<br>  | • Zero Downtime Migration (ZDM) - Online or offline migration using OCI Object Storage<br>• Oracle Data Pump<br>• Oracle SQL*Loader | • Logical Migrations with Zero Downtime Migration<br>• Load Data Using Oracle Data Pump<br>• Load Data Using SQL*Loader |
| Autonomous Database Serverless → ADB-D on Public Cloud or Exadata Cloud@Customer<br>  | • Oracle GoldenGate Extract configured to capture from the source Autonomous Database.<br>• Oracle Data Pump | • Use Oracle GoldenGate to Replicate Data with Autonomous Database<br>• 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<br>  | • Oracle Data Pump<br>• Zero Downtime Migration (ZDM 21.2) supports RDS and AWS | • Load Data Using Oracle Data Pump<br>• Logical Migrations with Zero Downtime Migration |
| Autonomous Database on Dedicated Exadata Infrastructure → Autonomous Database on Dedicated Exadata Infrastructure<br>  | • Oracle GoldenGate Replicat configured to apply to the target Autonomous Database.<br>• Oracle Data Pump | • Use Oracle GoldenGate to Replicate Data with Autonomous Database<br>• Load Data Using Oracle Data Pump |

> **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 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 Database on Dedicated Exadata Infrastructure

You can bulk load data into Autonomous 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 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

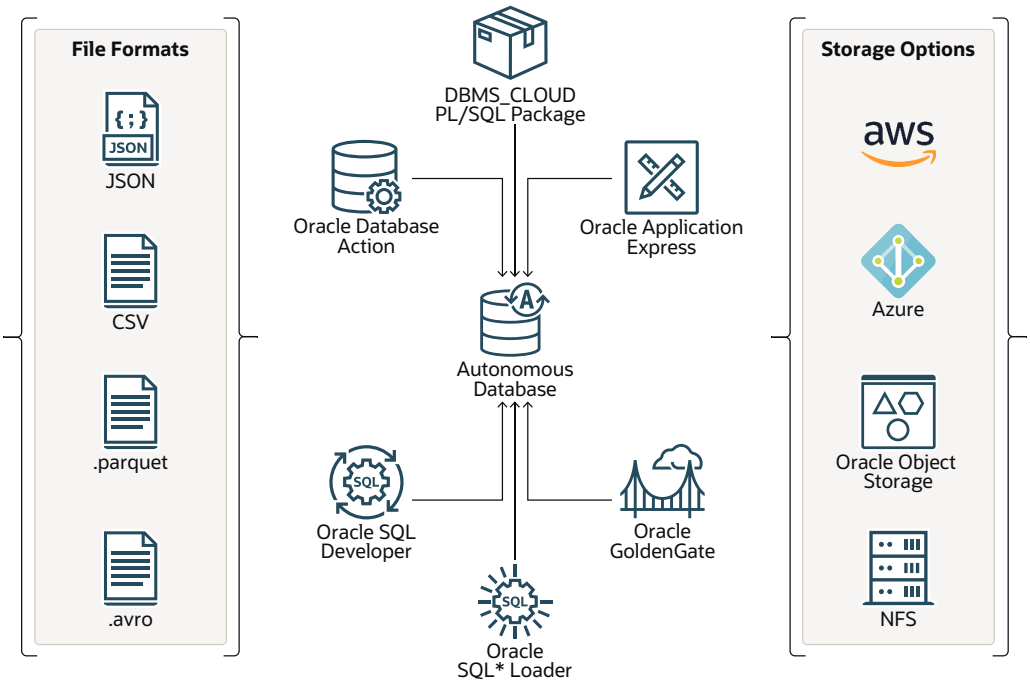
Migrating Data into Autonomous Database to explore the different data migration options supported by Autonomous Database.

## 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 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 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 Database on Public Cloud | Autonomous 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 Database.<br>You can load data from different file formats using the `DBMS_CLOUD.COPY_DATA` and the `DBMS_CLOUD.COPY_TEXT` procedures.<br>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<br>Attach Network File Storage to Autonomous Database<br>Attach External File Storage to Autonomous Database |
| Database Actions | Web-based interface with development, data tools, administration, and monitoring features in Autonomous Database.<br>Database Actions is bundled with each Autonomous Database instance. | ✔ | ✔ | Small test and development datasets.<br>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 Database on Public Cloud | Autonomous 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 Database on Dedicated Exadata Infrastructure.<br>One time bulk data loading from cloud storage.<br>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.<br>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.<br>You can then move these databases to Autonomous Database using PL/SQL subprograms. | ✔ | ✔ | Small test and development datasets.<br>Adopting Oracle Autonomous Database. | Database Export (Unload Database Objects and Data)<br>Deploy Objects to Cloud |

| Tool/Utility | Description | Autonomous Database on Public Cloud | Autonomous 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.<br>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 Database on Dedicated Exadata Infrastructure.<br>Scalable active-active architecture.<br>Real-time data warehouse.<br>Big data integration.<br>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 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 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_CREDENTIAL.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 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 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 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.

## 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
UPDATE_TIME                           LOGFILE_TABLE   BADFILE_TABLE
---------- ----------- ------- ---------- ----------------------
---------------------- --------------- ------------- ------------- -------------
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.

## Use Oracle Data Pump

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

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

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

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

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

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

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

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

## Upload the Export Files to Cloud Object Storage

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

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

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

## Import Data Using Oracle Data Pump

Oracle recommends using the latest Oracle Data Pump version for importing data from Data Pump files into your Autonomous 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 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. 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.

2. If you are using Oracle Data Pump version 12.2.0.1 or earlier, set the credential as the default credential for your Autonomous 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://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/
idthydc0kinr/mybucket/export%u.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%u.dmp \
     parallel=16 \
     transform=segment_attributes:n \
     exclude=cluster, db_link
```

   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.

> **Note:**
>
> To perform a full import or to import objects that are owned by other users, you need the `DATAPUMP_CLOUD_IMP` role.

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

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

### 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 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 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 on SQL*Loader parameters see *Oracle Database Utilities*.

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

Autonomous 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.

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

## Migrate to Autonomous Database Using Oracle Database Migration Service

You can migrate data to an Autonomous 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 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 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 Database Serverless and Autonomous 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 Database

You can replicate data to Autonomous Database using Oracle GoldenGate and Oracle GoldenGate Cloud Service. You can capture changes from the Autonomous Database and replicate to any target database or platform that Oracle GoldenGate supports, including other Autonomous Database on Dedicated Exadata Infrastructures.

Oracle GoldenGate offers efficient data replication solutions in multiple scenarios, some of which are listed below:

- Data migration to or between Autonomous 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 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 Database. This account is used for both Extracts and Replicats in the Autonomous 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 Database:** This is required only if you want to use Autonomous Database as the source database in the data replication activity. For complete details, see Configuring Extract to Capture from an Autonomous Database.

> **Note:**
>
> GoldenGate capture support is available on autonomous container databases running version 19.11 or higher.

- **Configure Replicat to apply to an Autonomous Database:** This is required only if you want to use Autonomous Database as the target database in the data

replication activity. For complete details, see Configuring Replicat to Apply to an Autonomous Database.
Parallel Replicat is a variant of Replicat that applies transactions in parallel to improve performance. Autonomous 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 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 Database using Oracle Goldengate replication, run *Lab 10: Oracle GoldenGate Replication* in the Oracle Autonomous Database Dedicated for Fleet Administrators Workshop.

# Export Data from Autonomous Database

## Use Oracle Data Pump to Export Data from Autonomous Database

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

Oracle recommends using the latest Oracle Data Pump version for exporting data from Autonomous Database to other Oracle databases, as it contains enhancements and fixes for a better experience. Download the latest version of Oracle Instant Client and download the Tools Package, which includes Oracle Data Pump, for your platform from Oracle Instant Client Downloads. See the installation instructions on the platform install download page for the installation steps required after you download Oracle Instant Client and the Tools Package.

To move data from an Autonomous 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 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 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 Database by exporting it to a directory on your database:

  1. Export data to a directory on Autonomous Database and move the dump file set from the directory to your Cloud Object Store. See Export Data Using a Directory on Autonomous 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 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 Database by exporting it directly to your object store:

  1. Export data from Autonomous Database to a Cloud Object Store directly. See Export Data From Autonomous 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 Database

To export data using a directory on Autonomous 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 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 Database named `ATPC1` with 16 OCPUs:

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

> **Note:**
>
> In case of ECPUs, `parallel` must be set to 0.25 times the ECPU count. The above example with 16 ECPUs will have `parallel` set to 0.25 x 16, that is, 4 with `expdp`.

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

For the best export performance use the `HIGH` database service for your export connection and set the `PARALLEL` parameter to the number of 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 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                    BYTES  CHECKSUM                    CREATED
LAST_MODIFIED
---------------------- ---------- --------- ------------------------------
--------------------
exp01.dmp                      12288               12-NOV-19 06.10.47.0 PM GMT       12-
NOV-19...
exp02.dmp                       8192               12-NOV-19 06.10.48.0 PM GMT       12-
NOV-19...
exp03.dmp                    1171456               12-NOV-19 06.10.48.0 PM GMT       12-
NOV-19...
exp04.dmp                     348160               12-NOV-19 06.10.48.0 PM GMT       12-
NOV-19...
export.log                      1663               12-NOV-19 06.10.50.0 PM GMT       12-
NOV-19...
```

> **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 on Oracle Data Pump Export see *Oracle Database Utilities*.

### Move Dump File Set from Autonomous Database to Your Cloud Object Store

1. Connect to your Autonomous 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 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 Database to Object Store Directly

Shows how to use Oracle Data Pump to export data from Autonomous Database to Cloud Object Store directly.

1. Connect to your Autonomous 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 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%u.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%u.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.

For the best export performance use the `high` database service for your export connection and set the `parallel` parameter to the number of OCPUs or 0.25 times the number of ECPUs in your Autonomous Database. For information on which database service name to connect to run Data Pump Export, see Predefined Database Service Names for Autonomous 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 Database, see Limitations on the Use of SQL Commands.

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

> **Note:**
>
> 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 Oracle Database Utilities for more details.

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 Database.

If you export directly to Object Store using Oracle Data Pump, as shown in Export Data From Autonomous 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 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 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 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 on Oracle Data Pump Import parameters see *Oracle Database 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 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 Database
- Attach External File Storage to Autonomous Database

## Export Data as CSV to a Directory

Shows the steps to export table data from your Autonomous Database to a directory as CSV data by specifying a query.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Create a directory.

   For example:

   ```
   CREATE DIRECTORY export_dir AS 'export_dir';
   ```

   See Create 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 Database to a directory as JSON data by specifying a query.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Create a directory.

   For example:

   ```
   CREATE DIRECTORY export_dir AS 'export_dir';
   ```

   See Create 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 XML to a Directory

Shows the steps to export table data from your Autonomous Database to Directory as XML data by specifying a query.

1. Connect to your Autonomous Database instance.

   See Connect to Autonomous Database for more information.

2. Create a directory.

   For example:

   ```
   CREATE DIRECTORY export_dir AS 'export_dir';
   ```

   See Create 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 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 Database to Cloud Object Storage as JSON data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database, and you can use an Oracle Cloud Infrastructure resource principal to access your Oracle Cloud Infrastructure Object Store, Amazon Resource Names (ARNs) to access AWS Simple Storage Service (S3), an Azure service principal to access Azure BLOB storage, or a Google service account to access Google Cloud Platform (GCP) resources.

1. Connect to your Autonomous Database instance.

See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
    password => 'password'
  );
END;
/
```

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

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, 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.
- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.

## Export Data as CSV to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as CSV data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database. 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 Database instance.

   See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

```
BEGIN
  DBMS_CLOUD.CREATE_CREDENTIAL(
    credential_name => 'DEF_CRED_NAME',
    username => 'user1@example.com',
```

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

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

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, 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.

- When you no longer need the files that you export, use the procedure `DBMS_CLOUD.DELETE_OBJECT` or use native Cloud Object Storage commands to delete the files.

## Export Data as XML to Cloud Object Storage

Shows the steps to export table data from your Autonomous Database to Cloud Object Storage as XML data by specifying a query.

This export method supports all the Cloud Object Stores supported by Autonomous Database. 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 Database instance.

   See Connect to Autonomous Database for more information.

2. Store your Cloud Object Storage credential using `DBMS_CLOUD.CREATE_CREDENTIAL`.

   For example:

   ```
   BEGIN
     DBMS_CLOUD.CREATE_CREDENTIAL(
       credential_name => 'DEF_CRED_NAME',
       username => 'user1@example.com',
       password => 'password'
     );
   END;
   /
   ```

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

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, 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.
- 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, or XML)

Describes the output file naming using `DBMS_CLOUD.EXPORT_DATA` with CSV, JSON, 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, or XML).

To speed up the procedure and to generate the output as fast as possible, `DBMS_CLOUD.EXPORT_DATA` divides its work. This means that, depending on system resources, when you run `DBMS_CLOUD.EXPORT_DATA` the procedure creates multiple output files either in the Cloud Object Store bucket or in the directory.

The format for each generated file is:

[*FileNamePrefix* |
*client_info_module_action*]*_sequenceNum_timestamp*.*format_extension*.
[*compression_extension*]

- *FileNamePrefix*: (optional) If a *FileNamePrefix* is supplied, `DBMS_CLOUD.EXPORT_DATA` uses the file name prefix to generate file names for the results. The *FileNamePrefix* is specified using the text supplied after the bucket or directory name in the `file_uri_list` parameter value.

  You cannot provide multiple values for the *FileNamePrefix* in the `file_uri_list`.
- *client_info_module_action*: If a file name prefix is not supplied with the `file_uri_list` parameter, `DBMS_CLOUD.EXPORT_DATA` uses the combination of *client_info*, application *module* and *action* as the file name prefix (when this information is available). The procedure obtains these names from the application

information for the database session that runs the query. See DBMS_APPLICATION_INFO for information on *client_info*, *module* name, and *action* name.

If a file name prefix is not supplied with the `file_uri_list` and the database session attributes are not available, `DBMS_CLOUD.EXPORT_DATA` uses the file name prefix "`data`".

- *sequenceNum*: The sequence number associated with the `DBMS_CLOUD.EXPORT_DATA` query. Depending on the query, the database service, and the number of ECPUs (OCPUs if your database uses OCPUs) there are one or more *sequenceNum*s. Also, depending on the size of the results, there are one or more output files for each *sequenceNum*.
- *timestamp*: Timestamp when the file is uploaded.
- *format_extension*: The default value depends on the `format type` value:
  - CSV format: `.csv`
  - JSON format: `.json`
  - XML format: `.xml`

  For more information, see the description for `format` option `fileextension` in DBMS_CLOUD Package Format Options for EXPORT_DATA.
- *compression_extension*: When you include the `format` parameter with the `compression` option with the value `gzip`, this is "`gz`".

  When the `format type` is `parquet`, the `compression` value `snappy` is also supported and is the default.

For example, the file name prefix in the following `DBMS_CLOUD.EXPORT_DATA` procedure is specified in the `file_uri_list` parameter, as `dept_export`. The example generates the output to the provided Object Store in the specified format.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-phoenix-1.oraclecloud.com/n/
namespace-string/b/bucketname/o/dept_export',
    query           => 'SELECT * FROM DEPT',
    format          => JSON_OBJECT('type' value 'json'));
END;
/
```

When you specify a file name prefix the generated output files include the file name prefix, similar to the following:

```
dept_export_1_20210809T173033Z.json
dept_export_2_20210809T173034Z.json
dept_export_3_20210809T173041Z.json
dept_export_4_20210809T173035Z.json
```

The number of generated output files depends on the size of the results, the database service, and the number of ECPUs (OCPUs if your database uses OCPUs) in the Autonomous Database instance.

In the following example the `file_uri_list` parameter does not include a file name prefix and the `compression` parameter is supplied, with value `gzip`:

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    credential_name => 'DEF_CRED_NAME',
    file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/',
    query           => 'SELECT * FROM DEPT',
    format          => json_object('type' value 'json', 'compression'
value 'gzip'));
END;
/
```

When a file name prefix is not in the `file_uri_list` parameter, `DBMS_CLOUD.EXPORT_DATA` uses a file name prefix of the form: *client_info_module_action*. For this example the generated output files include the file name prefix that `DBMS_CLOUD.EXPORT_DATA` supplies and the files are compressed with gzip and the file extension `.gz` is added, as follows:

```
Client1_Module1_Action1_1_20210809T173033Z.json.gz
Client1_Module1_Action1_2_20210809T173034Z.json.gz
Client1_Module1_Action1_3_20210809T173041Z.json.gz
Client1_Module1_Action1_4_20210809T173035Z.json.gz
```

If the *client_info_module_action* session information is not available when you run `DBMS_CLOUD.EXPORT_DATA`, the file name prefix is set to `data`. For example:

```
data_1_20210809T173033Z.json.gz
data_2_20210809T173034Z.json.gz
data_3_20210809T173041Z.json.gz
data_4_20210809T173035Z.json.gz
```

For example, the file name prefix in the following `DBMS_CLOUD.EXPORT_DATA` procedure is specified in the `file_uri_list` parameter, as `dept_export`. The example generates the output to the provided directory in the specified format.

```
BEGIN
  DBMS_CLOUD.EXPORT_DATA(
    file_uri_list   => 'DATA_PUMP_DIR:sales.json',
    query           => 'SELECT * FROM SALES',
    format          => JSON_OBJECT('type' value 'json'));
END;
/
```

When you specify a file name prefix the generated output file include the file name prefix, similar to the following:

```
sales_1_20230705T124523275915Z.csv
```

Notes for file naming with `DBMS_CLOUD.EXPORT_DATA`:

- `DBMS_CLOUD.EXPORT_DATA` does not create buckets or directories.
- The number of files that `DBMS_CLOUD.EXPORT_DATA` generates is determined by the number of ECPUs (OCPUs if your database uses OCPUs), the database service, and the size of the result data.
- The following applies when providing a directory object name in the `file_uri_list` parameter:
  - The provided directory must exist and you must have `WRITE` access to the directory.
  - The directory name is case-sensitive when enclosed in double quotes.
  - The credential name parameter must not be provided.
- For CSV, JSON, or XML output, by default when a generated file contains 10MB of data a new output file is created. However, if you have less than 10MB of result data you may have multiple output files, depending on the database service and the number of ECPUs (OCPUs if your database uses OCPUs) for the Autonomous Database instance.

  The default output file chunk size is 10MB for CSV, JSON, or XML. You can change this value with the `format` parameter `maxfilesize` option. See DBMS_CLOUD Package Format Options for EXPORT_DATA for more information.

# Manage Users

## Managing Database Users

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

### Create Database Users

You can create database users in Autonomous 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 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 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 Database.

Autonomous 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 Database whose ADMIN user password you want to change.

   For instructions, see View Details of a Dedicated Autonomous 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 Database come with a predefined database role named `DWROLE`. 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.

## 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 Database

You can create and alter user profiles in Autonomous Database. After you create or alter a profile, you can specify the profile clause with `CREATE USER` or `ALTER USER`. You can also import existing user profiles from another environment with Oracle Data Pump Import.

> **Note:**
>
> Autonomous Database has restrictions on the profile clause. See 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;
```

> **Note:**
>
> If you are not the ADMIN user, then you must have `CREATE PROFILE` privilege to run `CREATE PROFILE`. If you run `ALTER PROFILE`, then you must have `ALTER PROFILE` privilege.

2. Use the new or altered profile with a `CREATE USER` or `ALTER USER` command. For example:

```
CREATE USER new_user IDENTIFIED BY password PROFILE new_profile;
GRANT CREATE SESSION TO new_user;
```

This creates *new_user* with profile *new_profile* and with connect privileges. The *new_user* can now connect to the database and run queries. To grant additional privileges to users, see Manage Database User Privileges.

See CREATE PROFILE for information on using `CREATE PROFILE` or `ALTER PROFILE`.

You can import existing profiles created in other environments using Oracle Data Pump Import (`impdp`). Any existing profile association with database users is preserved after importing into Autonomous Database. When a newly created user, created from an Oracle Data Pump import, attempts to login for the first time, the login is handled as follows:

- The password complexity restrictions are the same as the restrictions for any user on Autonomous Database.
- If the user's password violates the password complexity requirements, the account is expired with a 30-day grace period. In this case the user is required to change their password before the grace period ends.

> **Note:**
>
> Profile assignments for users with profile `ORA_PROTECTED_PROFILE` cannot be modified.

When you create or alter a profile, you can specify a Password Verification Function (PVF) to manage password complexity. See Manage Password Complexity on Autonomous Database for more information.

## Manage Password Complexity on Autonomous Database

You can create a Password Verify Function (PVF) and associate the PVF with a profile to manage the complexity of user passwords.

> **Note:**
>
> The minimum password length for a user specified PVF is 8 characters and must include at least one upper case letter, one lower case letter and one numeric character. The minimum password length for the DEFAULT profile is 12 characters (the DEFAULT profile uses the `CLOUD_VERIFY_FUNCTION` PVF). The password cannot contain the username.
> Oracle recommends using a minimum password length of 12 characters. If you define a profile's PVF, and set the minimum password length to less than 12 characters, then tools such as Oracle Database Security Assessment Tool (DBSAT) and Qualys report this as a database security risk.

For example, to specify a PVF for a profile, use the following command:

```
CREATE PROFILE example_profile LIMIT PASSWORD_VERIFY_FUNCTION
ADMIN.EXAMPLE_PVF
```

If the profile is created or altered by any user other than the ADMIN user, then you must grant the `EXECUTE` privilege on the PVF. If you create a PVF and the password check fails, the database reports the `ORA-28219` error.

You can specify an Oracle supplied PVF, from one of the following:

- `CLOUD_VERIFY_FUNCTION` (this is the default password verification function for Autonomous Database):
  This function checks for the following requirements when users create or modify passwords:
  - The password must be between 12 and 30 characters long and must include at least one uppercase letter, one lowercase letter, and one numeric character.
  - The password cannot contain the username.
  - The password cannot be one of the last four passwords used for the same username.
  - The password cannot contain the double quote (") character.
  - The password must not be the same password that is set less than 24 hours ago.
- `ORA12C_STIG_VERIFY_FUNCTION`
  This function checks for the following requirements when users create or modify passwords:
  - The password has at least 15 characters.
  - The password has at least 1 lower case character and at least 1 upper case character.
  - The password has at least 1 digit.
  - The password has at least 1 special character.
  - The password differs from the previous password by at least 8 characters.

  See ora12c_stig_verify_function Password Requirements for more information.

Note the following restrictions for a Password Verification Function (PVF) that you create and assign to a profile:

- If you specify a user profile, the minimum password length depends on how you define the associated PVF, as follows:
  - If a PVF is defined, then the minimum password length enforced is 8 characters with at least one upper case letter, one lower case letter and one numeric character. The password cannot contain the username.
  - If the PVF is defined as `NULL`, then the minimum password length enforced is 8 characters with at least one upper case letter, one lower case letter and one numeric character. The password cannot contain the username.
  - If the profile does not have a PVF defined, then the DEFAULT profile's PVF (`CLOUD_VERIFY_FUNCTION`) is assigned and the minimum password length enforced is 12 characters.

- If you specify a Password Verify Function (PVF) that is more strict than the default `CLOUD_VERIFY_FUNCTION`, then the new verify function is used.
- A PVF that you create must be created as a `DEFINER RIGHTS` PL/SQL function. If a `INVOKER` rights PVF is provided as input to `CREATE` or `ALTER PROFILE`, then `ORA-28220` error is thrown.
- Any PVF that you create must be created in the ADMIN user schema. If a non-ADMIN user owned PVF is provided as input to `CREATE` or `ALTER PROFILE`, then `ORA-28220` error is thrown.
- A PVF cannot be altered or dropped by a non-ADMIN user. That is, any user with the `CREATE` or `DROP ANY PROCEDURE` privilege is not allowed to alter or drop a PVF.
- If the PVF associated with a profile is dropped, then any attempt to change the password for a user who uses the PVF in their profile throws the error `ORA-7443`. Users can still login when the PVF associated with their profile is dropped. However, if a user's password is expired and the PVF is dropped, then the user cannot login.

  To recover from the `ORA-7443` error, the ADMIN user must recreate the dropped PVF and assign it to the profile, or assign an existing PVF to the profile. This allows a user change their password and login.
- The `CREATE ANY PROCEDURE` system privilege and `DROP ANY PROCEDURE` system privilege are audited for PVF security. See the `PROCEDURES` list in Listings of System and Object Privileges for more information.

See Managing the Complexity of Passwords for more information.

## Gradual Database Password Rollover for Applications

An application can change its database passwords without an administrator having to schedule downtime.

To accomplish this, you can associate a profile having a non-zero limit for the `PASSWORD_ROLLOVER_TIME` password profile parameter, with an application schema. This allows the database password of the application user to be altered while allowing the older password to remain valid for the time specified by the `PASSWORD_ROLLOVER_TIME` limit. During the rollover period of time, the application instance can use either the old password or the new password to connect to the database server. When the rollover time expires, only the new password is allowed.

See Managing Gradual Database Password Rollover for Applications for more information.

# Use Oracle Database Vault to Manage Database User Privileges

Autonomous 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 to gain an understanding of the impact of configuring and enabling Database Vault.

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

> **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 Database Dedicated for Security Administrators Workshop.

## Configure and Enable Oracle Database Vault

To configure and enable Enable Oracle Database Vault in your Autonomous 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 Database on Dedicated Exadata Infrastructure deployment.

# Use Identity and Access Management (IAM) Authentication with Autonomous Database

APPLIES TO: ● Oracle Public Cloud only

You can configure Autonomous Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous Database with IAM credentials.

> **Note:**
>
> Autonomous Database integration with Oracle Cloud Infrastructure (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 Database supports users and groups in default and non-default identity domains.

## About Identity and Access Management (IAM) Authentication with Autonomous Database

Oracle Cloud Infrastructure IAM integration with Autonomous Database supports both database password authentication and token-based authentication. See Authenticating and Authorizing IAM Users for Oracle Autonomous Databases for complete details about the architecture for using IAM users on Autonomous Database.

**IAM Database Password Authentication**

> **Note:**
>
> Any supported 12c and above database client can be used for IAM database password access to Autonomous Database.

An Oracle Cloud Infrastructure IAM database password allows an IAM user to log in to an Autonomous Database instance as Oracle Database users typically log in with a user name and password. The user enters their IAM user name and IAM database password. An IAM database password is a different password than the Oracle Cloud Infrastructure Console password. Using an IAM user with the password verifier you can login to Autonomous Database with any supported database client.

**Identity and Access Management (IAM) SSO Token Based Authentication**

There are several ways a database client can obtain an IAM database token:

- A client application or tool can request the database token from IAM for the user and can pass the database token through the client API. Using the API to send the token overrides other settings in the database client. This type of IAM database token usage is supported for the following clients:
  - JDBC-thin on all platforms
  - Oracle Instant Client OCI-C on Linux
  - Oracle Data Provider for .NET (ODP.NET) Core
- If the application or tool does not support requesting an IAM database token through the client API, the IAM user can first use Oracle Cloud Infrastructure command line interface (CLI) to retrieve the IAM database token and save it in a file location. For example, to use SQL*Plus and other applications and tools using this connection method, you first obtain the database token using the Oracle Cloud Infrastructure (OCI) Command Line Interface (CLI). If the database client is configured for IAM database tokens, when a user logs in

with the slash login form, the database driver uses the IAM database token that has been saved in a default or specified file location.

- A client application or tool can use an Oracle Cloud Infrastructure IAM instance principal or resource principal to get an IAM database token, and use the IAM database token to authenticate itself to an Autonomous Database instance.
- IAM users and OCI applications can request a database token from IAM with several methods, including using an API-key. See Configuring a Client Connection for SQL*Plus That Uses an IAM Token for an example. See About Authenticating and Authorizing IAM Users for an Oracle Autonomous Database for a description of other methods such as using a delegation token within an OCI cloud shell.

If a user enters a username/password to login, then the database driver uses the password verifier method to access the database regardless of the client's database token setting.

## Prerequisites for Identity and Access Management (IAM) Authentication on Autonomous Database

Before using Identity and Access Management (IAM) authentication on Autonomous Database, you must 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 Database resources reside.

1. Create a service gateway in the VCN (Virtual Cloud Network) where your Autonomous 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 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 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:
      - **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.

## Change External Identity Providers on Autonomous 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 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 Database. However, there can only be one external authentication scheme enabled at any given time.

To enable database users to connect to Autonomous 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 Database
   - Enable Kerberos Authentication on Autonomous Database
   - Configure the Oracle Autonomous Database for Microsoft Azure AD Integration
3. Re-enable IAM users to connect to Autonomous 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 Database instances.

A policy is a group of statements that specifies who can access particular resources, and how. Access can be granted for the entire tenancy, databases in a compartment, or individual databases. This means you write a policy statement that gives a specific group a specific type of access to a specific type of resource within a specific compartment.

> **Note:**
>
> Defining a policy is required to use IAM tokens to access Autonomous Database. A policy is not required when using IAM database passwords to access Autonomous Database.

To enable Autonomous Database to allow IAM users to connect to the database using IAM tokens:

1. Perform Oracle Cloud Infrastructure Identity and Access Management prerequisites by creating a group and adding users to the group.

   For example, create the group `sales_dbusers`.

   See Managing Groups for more information.

2. Write policy statements to enable access to Oracle Cloud Infrastructure resources.

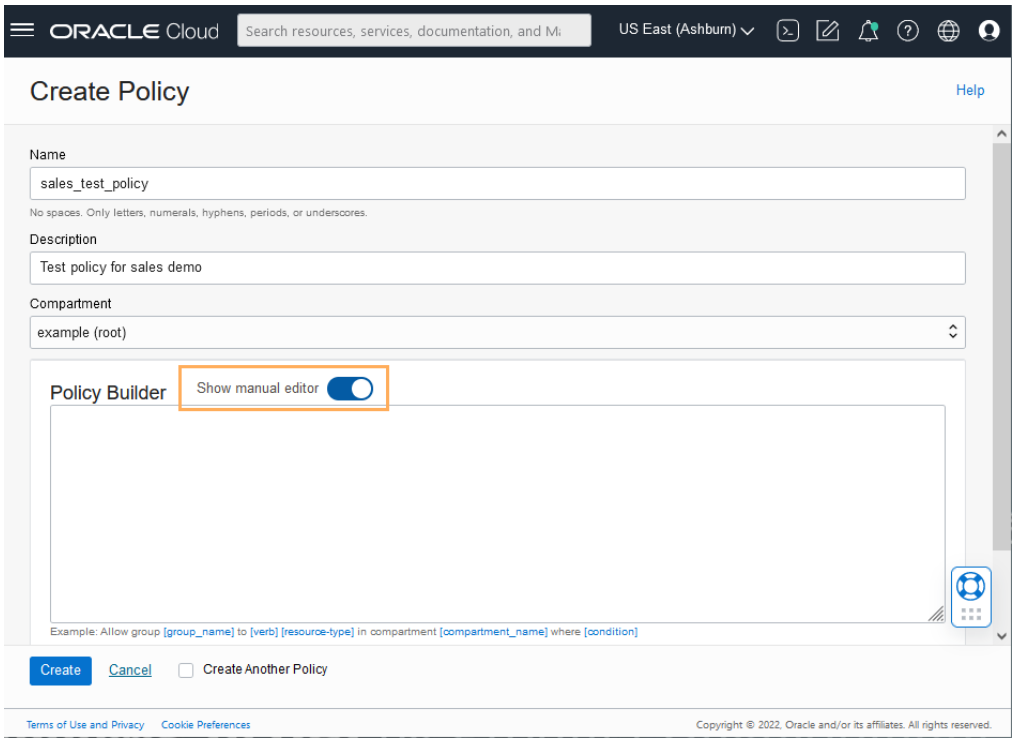
   a. In the Oracle Cloud Infrastructure console click **Identity & Security**.

   b. Under **Identity & Security** click **Policies**.

   c. To a write policy, click **Create Policy**.

   d. On the Create Policy page, enter a Name and a Description.

   e. On the Create Policy page, select **Show manual editor**.




   f. Use the **Policy Builder** to create a policy.

   For example to create a policy to allow users in IAM group `DBUsers` to access any Autonomous Database in their tenancy:

   ```
   Allow group DBUsers to use autonomous-database-family in tenancy
   ```

   For example to create a policy that limits members of `DBUsers` group to access Autonomous Databases in compartment `testing_compartment` only:

   ```
   allow group DBUsers to use autonomous-database-family in compartment
   testing_compartment
   ```

For example to create a policy that limits group access to a single database in a compartment:

```
allow group DBUsers to use autonomous-database-family in
compartment testing_compartment where target.database.id =
'ocid1.autonomousdatabase.oc1.iad.aaaabbbbcccc'
```

Refer to Creating an IAM Policy to Authorize Users Authenticating with Tokens in *Database Security Guide* for more information on IAM policies to access the database.

g. Click **Create**.

See Managing Policies for more information on policies.

Notes for creating policies for use with IAM users on Autonomous Database:

- Policies can allow IAM users to access Autonomous Database instances across the entire tenancy, in a compartment, or can limit access to a single Autonomous Database instance.
- You can use either instance principal or resource principal to retrieve database tokens to establish a connection from your application to an Autonomous Database instance. If you are using an instance principal or resource principal, you must map a dynamic group. Thus, you cannot exclusively map instance and resource principals; you only can map them through a shared mapping and putting the instance or resource instance in an IAM dynamic group.

  You can create Dynamic Groups and reference dynamic groups in the policies you create to access Oracle Cloud Infrastructure. See Plan Access Controls and Create Supporting Resources and Managing Dynamic Groups for details.

## Add IAM Users on Autonomous Database

To add IAM users to allow access to Autonomous Database, map database global users to IAM groups or users with `CREATE USER` or `ALTER USER` statements (with `IDENTIFIED GLOBALLY AS` clause).

The authorization of IAM users to an Autonomous Database instance works by mapping IAM global users (schemas) to IAM users (exclusive mapping) or IAM groups (shared schema mapping).

**To authorize IAM users on an Autonomous Database instance:**

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).
2. Create a mapping between the Autonomous Database user (schema) with `CREATE USER` or `ALTER USER` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the IAM group name.

   Use the following syntax to map a global user to an IAM group:

   ```
   CREATE USER global_user IDENTIFIED GLOBALLY AS
   'IAM_GROUP_NAME=IAM_GROUP_NAME';
   ```

For example, to map an IAM group named `db_sales_group` to a shared database global user named `sales_group`:

```
CREATE USER sales_group IDENTIFIED GLOBALLY AS
     'IAM_GROUP_NAME=db_sales_group';
```

This creates a shared global user mapping. The mapping, with global user `sales_group`, is effective for all users in the IAM group. Thus, anyone in the `db_sales_group` can log in to the database using their IAM credentials (through the shared mapping of the `sales_group` global user).

The following example shows how to accomplish this for a non-default domain:

```
CREATE USER shared_sales_schema IDENTIFIED GLOBALLY AS
     'IAM_GROUP_NAME=sales_domain/db_sales_group';
```

3. If you want to create additional global user mappings for other IAM groups or users, follow these steps for each IAM group or user.

> **Note:**
>
> Database users that are not `IDENTIFIED GLOBALLY` can continue to login as before, even when the Autonomous Database is enabled for IAM authentication.

**To exclusively map a local IAM user to an Oracle Database Global User**:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).
2. Create a mapping between the Autonomous Database user (schema) with `CREATE USER` or `ALTER USER` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the IAM local IAM user name.

   For example, to create a new database global user named `peter_fitch` and map this user to an existing local IAM user named `peterfitch`:

   ```
   CREATE USER peter_fitch IDENTIFIED GLOBALLY AS
   'IAM_PRINCIPAL_NAME=peterfitch'
   ```

   The following example shows how to create the user by specifying a non-default domain, `sales_domain`:

   ```
   CREATE USER peter_fitch2 IDENTIFIED GLOBALLY AS
   'IAM_PRINCIPAL_NAME=sales_domain/peterfitch';
   ```

## Add IAM Roles on Autonomous Database

Optionally, create global roles to provide additional database roles and privileges to IAM users when multiple IAM users are mapped to the same shared global user.

The use of global roles is optional when using either an exclusive IAM mapping to user (schema) or shared user mapping on Autonomous Database. For example, all privileges and roles can be granted to the shared schema and all IAM users who map to the shared schema would be granted the privileges and roles assigned to the shared schema.

You can use a global role to optionally differentiate users who use the same shared schema. For example, a set of users can all have the same shared schema and the shared schema could have the `CREATE SESSION` privilege. Then global roles can be used to provide differentiated privileges and roles assigned to different groups of users who all use the same shared schema.

Granting additional roles to IAM users in Autonomous Database works by mapping Autonomous Database global roles to IAM groups.

To map Autonomous Database global roles to IAM groups:

1. Log in as the ADMIN user to the database that is enabled to use IAM (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).
2. Set database authorization for Autonomous Database roles with `CREATE ROLE` or `ALTER ROLE` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the IAM group name.

   Use the following syntax to map a global role to an IAM group:

   ```
   CREATE ROLE global_role IDENTIFIED GLOBALLY AS
       'IAM_GROUP_NAME=IAM_GROUP_of_WHICH_the_IAM_USER_IS_a_MEMBER';
   ```

   For example, to map an IAM group named `ExporterGroup` to a shared database global role named `export_role`:

   ```
   CREATE ROLE export_role IDENTIFIED GLOBALLY AS
        'IAM_GROUP_NAME=ExporterGroup';
   ```

   The following example shows how to create the role by specifying a non-default domain, `sales_domain`:

   ```
   CREATE ROLE export_role IDENTIFIED GLOBALLY AS
        'IAM_GROUP_NAME=sales_domain/ExporterGroup';
   ```

   All members of the `ExporterGroup` in the `sales_domain` domain will be authorized with the database global role `export_role` when they log in to the database.

3. Use `GRANT` statements to grant the required privileges or other roles to the global role.

```
GRANT CREATE SESSION TO export_role;
GRANT DWROLE TO export_role;
```

4. If you want an existing database role to be associated with an IAM group, then use `ALTER ROLE` statement to alter the existing database role to map the role to an IAM group. Use the following syntax to alter an existing database role to map it to an IAM group:

```
ALTER ROLE existing_database_role
   IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=IAM_Group_Name';
```

If you want to add additional global role mappings for other IAM groups, follow these steps for each IAM group.

## Create IAM Database Password for IAM Users

To add an IAM user and allow the IAM user to login to Autonomous Database by supplying a username and password, you must create an IAM database password.

See Working with IAM Database Passwords for more information.

## Connect to Autonomous Database with Identity and Access Management (IAM) Authentication

After the database ADMIN user maps global users and global roles to the IAM users and IAM groups, users log in to the Autonomous Database instance using their Oracle Cloud Infrastructure IAM credentials or access the database through an Oracle Cloud Infrastructure IAM database token.

You can still log in to the Autonomous Database using your local database account username and password (non-global database user account).

You can use a database client to access an Autonomous Database instance as an Oracle Cloud Infrastructure IAM user. To use a client with Oracle Cloud Infrastructure IAM username and password credentials and a password verifier, the database client must be 12c or newer.

Alternatively, you can use an Oracle Cloud Infrastructure IAM database token to access an Autonomous Database instance 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 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 Database on Dedicated Exadata Infrastructure Instance Using IAM

IAM users can connect to the Autonomous 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 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 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 Database on Dedicated Exadata Infrastructure instance.

### Client Connections That Use a Token

For IAM token access to the Autonomous 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 Database instance by using the following syntax:

  ```
  CONNECT user_name@db_connect_string
  Enter password: password
  ```

  In this specification, *`user_name`* is the IAM user name. There is a limit of 128 bytes for the combined *`domain_name/user_name`*.

The following example shows how IAM user `peter_fitch` can log in to an Autonomous Database instance.

```
sqlplus /nolog
connect peter_fitch@db_connect_string
Enter password: password
```

Some special characters will require double quotation marks around *user_name* and *password*. For example:

```
"peter_fitch@example.com"@db_connect_string

"IAM database password"
```

## Configuring a Client Connection for SQL*Plus That Uses an IAM Token

You can configure a client connection for SQL*Plus that uses an IAM token.

1. Ensure you have an IAM user account.
2. Check with an IAM administrator and an Oracle Database administrator to ensure you have a policy allowing you to access the database in the compartment or your tenancy and that you are mapped to a global schema in the database.
3. If your application or tool does not support direct IAM integration, then download, install, and configure the OCI CLI. (See *OCI Command Line Interface Quickstart*.) Set up an API key as part of the OCI CLI configuration and select default values.
   a. Set up the API key access for the IAM user.
   b. Retrieve the `db-token`. For example:
      - Retrieving a `db-token` with an `API-key` using the Oracle Cloud Infrastructure (OCI) command-line interface:

        ```
        oci iam db-token get
        ```

      - Retrieving a `db-token` with a security (or session) token:

        ```
        oci iam db-token get --auth security_token
        ```

        If the security token has expired, a window will appear so the user can log in to OCI again. This generates the security token for the user. OCI CLI will use this refreshed token to get the `db-token`.

      - Retrieving a `db-token` with a delegation token: When you log in to the cloud shell, the delegation token is automatically generated and placed in the `/etc` directory. To get this token, 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 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.c
om))
  (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 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 Database. If an IAM user name and IAM database password are specified instead of slash `/`, then the database client will connect using the password instead of using the `db-token`.

## Use Instance Principal to Access Autonomous Database with Identity and Access Management (IAM) Authentication

After the ADMIN user enables Oracle Cloud Infrastructure IAM on Autonomous Database, an application can access the database through an Oracle Cloud Infrastructure IAM database token using an instance principal.

See Accessing the Oracle Cloud Infrastructure API Using Instance Principals for more information.

## Configure Proxy Authentication

Proxy authentication allows an IAM user to proxy to a database schema for tasks such as application maintenance.

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 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 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 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 Database as an Oracle Cloud Infrastructure IAM user.

> **Note:**
>
> Current user database link is not supported for connecting to an Autonomous 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 Database

Describes the steps to disable IAM external authentication user access for Autonomous Database.

You can disable IAM user access on your Autonomous Database instance 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 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 Database Tools with Identity and Access Management (IAM) Authentication

Provides notes for using Autonomous Database tools with IAM authentication enabled.

- Oracle Application Express is not supported for IAM users with Autonomous Database. See Access Oracle Application Express Administration Services for information on using regular database users with Autonomous Database.
- Database Actions is not supported for IAM users with Autonomous Database. See Enabling User Access to Database Actions for information on using regular database users with Autonomous Database.

# About Kerberos Authentication

You can configure Autonomous 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 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 Database uses these values for various components that make up a service principal's name:

| Component of Service Principal | Value in Autonomous Database |
|---|---|
| `kinstance` | The value of the `'host'` parameter found in the TNS connect string. |
| `kservice` | You can retrieve `GUID` for your Autonomous Database using a query on the `GUID` column of `V$PDBS`. It must be in uppercase when used as the value of `kservice`. |
| REALM | Any realm supported by your KDC. `REALM` must always be in uppercase. |

To enable Kerberos authentication for your Autonomous Database, you must keep your Kerberos configuration files (`krb.conf`) and service key table file (`v5srvtab`) ready. For more information on these files and steps to obtain them, please see About Kerberos Configuration Files.

## Notes About Kerberos Authentication on Autonomous Database

Before proceeding with Kerberos authentication on Autonomous Database on Dedicated Exadata Infrastructure, review the following notes:

- You can use Kerberos authentication only with Autonomous Database versions 19.18 or later.
- If you enable Kerberos authentication for your Autonomous Database, you can still use password-based database authentication for your database.
- Only one external authentication method can be used for your Autonomous 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 only with TCPS protocol.
- Kerberos authentication is not supported with DB_LINKs and databases with Autonomous Data Guard.

## Enable Kerberos Authentication on Autonomous Database

Follow the below-listed steps to enable Kerberos authentication for Autonomous Database on Dedicated Exadata Infrastructure:

- As there can only be one external authentication scheme enabled for an Autonomous 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.
- 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 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.

- After you enable Kerberos authentication, remove the configuration `krb.conf` and `v5srvtab` from Object Store. You can use local Object Store methods to remove these files or use `DBMS_CLOUD.DELETE_OBJECT` to delete the files from Object Store.

See Navigate to Oracle Cloud Infrastructure Object Storage and Create Bucket for more information on Object Storage.

See ENABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

## Disable Kerberos Authentication on Autonomous Database

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 Database

APPLIES TO: Oracle Public Cloud only

An Oracle Autonomous Database on Dedicated Exadata Infrastructure instance can be configured for Microsoft Azure AD users to connect using Azure `OAuth2` access tokens.

# About Authorizing Microsoft Azure AD Users for an Oracle Autonomous Database on Dedicated Exadata Infrastructure

Users for Oracle Autonomous Database on Dedicated Exadata Infrastructure can be centrally managed in a Microsoft Azure Active Directory (Azure AD) service.

You can perform this integration in the following Oracle Database environments:

- On-premises Oracle Database release 19.18 and later, but not for Oracle Database 21c
- Oracle Base Database Service
- Oracle Exadata Cloud Service (Oracle ExaCS)

The instructions for configuring Azure AD use the term "Oracle Database" to encompass these environments.

This type of integration enables the Azure AD user to access an Oracle Autonomous Database 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 administrator creates and configures the application registration (app registration) of the Oracle Autonomous Database on Dedicated Exadata Infrastructure instance with Azure AD. The database administrator also creates application (app) roles for the database app registration in Azure AD, and assigns these roles to Azure AD users, groups, and applications. These app roles will be mapped to the database global schemas and global roles. An Azure AD principal that is assigned to an app role will be mapped to either a database global schema or database global role. An Oracle global schema can also be mapped exclusively to an Azure AD user. When the principal is a guest user or service principal, they can only be mapped to the database schema through an Azure app role. An Oracle global role can only be mapped to an Azure app role.

Oracle Autonomous Database 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 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 token is a bearer token with an expiration time. The Oracle Database client driver will ensure that the token is in a valid format and that it has not expired before passing it to the database. The token is scoped for the database. Assigned app roles for the Azure AD principal are included as part of the access token. The directory location for the Azure AD token should only have enough permission for the user to write the token file to the location and the database client to retrieve these files (for example, just read and write by the process user). Because the token allows access to the database, it should be protected within the file system.

Azure AD users can request a token as a client registered with Azure AD app registration by using methods such as the following:

- Passing the Azure AD user name and password through a command line, script, file, or any other supported method

- Entering the Azure AD credentials into an Azure AD authentication screen with or without multi-factor authentication

Oracle Autonomous Database on Dedicated Exadata Infrastructure supports the following Azure AD authentication flows:

- Resource owner password credential (ROPC), which is used in non-graphic user interface environments when a pop-up window cannot be used to authenticate a user.
- Authorization code, which is used when a browser can be used to enter credentials for the user
- Client credentials, which are for applications that connect as themselves (and not the end-user)
- On-Behalf-Of (OBO), where an application requests an access token on behalf of a logged-in user to send to the database

Oracle Autonomous Database 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)

#### Architecture of the Microsoft Azure AD Integration with an Autonomous 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 Database

Microsoft Azure users must be mapped to an Autonomous Database schema and have the necessary privileges (through roles) before being able to authenticate to the Autonomous 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 Database Using Azure AD

Oracle Database supports three types of use cases for connecting to an Autonomous 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 Integrating Microsoft Azure AD with Oracle Autonomous Database on Dedicated Exadata Infrastructure

Both the Oracle and the Microsoft Azure administrators play roles in configuring the connection between Oracle Autonomous Database on Dedicated Exadata Infrastructure and Microsoft Azure AD.

The general process is as follows:

1. The Oracle administrator registers the database instance with the Microsoft Azure AD tenancy and then configures the Oracle Autonomous Database on Dedicated Exadata Infrastructure instance to connect with Azure AD as the identity service.
   As part of the registration process, the Oracle administrator or the Azure administrator creates or designates Azure app roles to be used for the mappings between the Oracle database and the Microsoft Azure endpoint.
2. The Oracle 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.
3. Optionally, the Oracle administrator creates and maps global Oracle Database roles to Azure app roles.
4. The Azure AD end user who wants to connect with the Oracle Autonomous 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.
5. 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 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 Database JDBC Developer's Guide*
   - Oracle Call Interface (OCI): *Oracle Call Interface Programmer's Guide*
   - Oracle Data Provider for .NET (ODP): *Oracle Data Provider for .NET Developer's Guide for Microsoft Windows*
6. Once connected to the Oracle Autonomous Database on Dedicated Exadata Infrastructure instance, the Azure AD end user performs tasks as needed.

## Configure the Oracle Autonomous Database on Dedicated Exadata Infrastructure for Microsoft Azure AD Integration

The Microsoft Azure AD integration with the Oracle Database requires the database to be registered with Azure AD so that the database can request the Azure AD public key.

**Prerequisites**

The Microsoft Azure AD integration with the Oracle Autonomous Database on Dedicated Exadata Infrastructure requires:

1. The Autonomous 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 Database to be registered with Azure AD.

**Procedure**

Implement the following tasks to configure your Autonomous Database for Microsoft Azure AD integration.

1. **Register the Autonomous Database instance with a Microsoft Azure AD tenancy:** Refer to Register the Oracle Autonomous Database Instance with a Microsoft Azure AD Tenancy in *Security Guide*.
2. **Manage app roles in Microsoft Azure AD:** Refer to Manage App Roles in Microsoft Azure AD in *Security Guide*.
3. **Configure outbound connectivity to Microsoft Azure AD using a NAT gateway:** Create a NAT gateway in the Virtual Cloud Network (VCN) where your Autonomous 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 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.
4. **Configure Azure AD as an external identity provider for Autonomous Database:**

By default, Autonomous 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 Database can enable only one external authentication scheme at a time.

To enable Azure AD as an external identity provider on an Autonomous Database instance:

a. Log in to the Autonomous Database instance as a user who has the `EXECUTE` privilege on the `DBMS_CLOUD_ADMIN` PL/SQL package. The ADMIN user has this privilege.

b. As there can only be one external authentication scheme enabled for an Autonomous 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 Database instance's Azure AD application registration.
  - `application_id`: Azure Application ID created in Azure AD to assign roles/ schema mappings for external authentication in the Autonomous Database instance.
  - `application_id_uri`: Unique URI assigned to the Azure Application.

This it the identifier for the Autonomous Database instance. The name must be domain qualified (this supports cross tenancy resource access).

The maximum length for this parameter is 256 characters.

- `force`: Set this parameter to `TRUE` if another `EXTERNAL AUTHENTICATION` method is configured for the Autonomous Database instance and you want to disable it.

For example:

```
BEGIN
  DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
      type   =>'AZURE_AD',
      params => JSON_OBJECT('tenant_id' VALUE
'29981886-6fb3-44e3-82',
                            'application_id' VALUE '11aa1a11-
aaa',
                            'application_id_uri' VALUE 'https://
example.com/111aa1aa'),
      force  => TRUE
  );
END;
```

This sets the `IDENTITY_PROVIDER_TYPE` system parameter.

For example, you can use the following to verify `IDENTITY_PROVIDER_TYPE`:

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE
NAME='identity_provider_type';

NAME                   VALUE
---------------------- --------
identity_provider_type AZURE_AD
```

See ENABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

## 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 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 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 Database

The connection between the Azure user, Azure AD, and the Autonomous 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 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.

## Trace Files Used for Troubleshooting Connections

You can generate two levels of trace files to troubleshoot Microsoft Azure AD connections on client side.

The two levels of trace files that you can generate are as follows:

- Low level tracing prints traces in case of failures:
  - If TCPS is not set up for the Azure AD connection, then it prints a message that the protocol has to be TCPS.
  - If `SSL_SERVER_DN_MATCH` is not set to `TRUE`, then it prints a message that the value is `FALSE`.
  - If `TOKEN_LOCATION` has not been specified, then it prints a message that the token location does not exist.
  - If the token is not present at the specified `TOKEN_LOCATION`, then it prints a message.
  - If the application has passed in the token without setting `OCI_ATTR_TOKEN_ISBEARER` to true, it prints a message for the missing attribute.
  - If the application has set `OCI_ATTR_TOKEN_ISBEARER` to `TRUE` and not passed in the token, it prints a message for the missing attribute.
  - If the token has expired, then it prints a message.
- High level tracing prints traces in case of failure as mentioned above. In addition, it prints traces in case of success, as follows:
  - It prints where `SSL_SERVER_DN_MATCH` is present, `tnsnames.ora` or `sqlnet.ora`. It also prints the value as `TRUE` if set to `TRUE`.
  - If both the token and `OCI_ATTR_TOKEN_ISBEARER=true` are set by the application, then it prints a message.
  - If `TOKEN_AUTH` has the correct value `OAUTH`, then it prints the value.
  - If the token is not expired, then it prints a message.

Refer to Setting Client Tracing for Token Authentication to learn how to add `EVENT` settings to the client-side `sqlnet.ora` file to control client tracing.

# Use Microsoft Active Directory with Autonomous Database on Dedicated Exadata Infrastructure

You can configure Autonomous Database on Dedicated Exadata Infrastructure to authenticate and authorize Microsoft Active Directory users. This configuration allows Active Directory users to access an Autonomous Database using their Active Directory credentials.

> **Note:**
>
> See Use Azure Active Directory (Azure AD) with Autonomous Database for information on using Azure Active Directory with Autonomous Database. The CMU option supports Microsoft Active Directory servers but does not support the Azure Active Directory service.

The integration of Autonomous Database with Centrally Managed Users (CMU) provides integration with Microsoft Active Directory. CMU with Active Directory works by mapping Oracle database global users and global roles to Microsoft Active Directory users and groups.

## Prerequisites to Configure CMU with Microsoft Active Directory on Autonomous Database

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

- You must have Microsoft Active Directory installed and configured. See AD DS Getting Started for more information.
- You must create an Oracle service directory user in Active Directory. See Connecting to Microsoft Active Directory for information on the Oracle service directory user account.
- An Active Directory system administrator must have installed Oracle password filter on the Active Directory servers, and set up Active Directory groups with Active Directory users to meet your requirements.

  Only password authentication is supported with CMU for Autonomous Database, so you must use the included utility, `opwdintg.exe`, to install the Oracle password filter on Active Directory, extend the schema, and create three new `ORA_VFR` groups for three types of password verifier generation. See Connecting to Microsoft Active Directory for information on installing the Oracle password filter.
- The Active Directory servers must be accessible from Autonomous Database through the public internet and the port 636 of the Active Directory servers must be open to Autonomous Database in Oracle Cloud Infrastructure, so that Autonomous Database can have secured LDAP access over TLS/SSL to the Active Directory servers through the internet.

  You can also extend your on-premise Active Directory to Oracle Cloud Infrastructure, where you can set up Read Only Domain Controllers (RODCs) for the on-premise Active Directory. Then you can use these RODCs in Oracle Cloud Infrastructure to authenticate and authorize the on-premise Active Directory users for access to Autonomous Databases.

  See 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 Database:
  - If you have configured CMU for an on-premise database, you can obtain these configuration files from your on-premise database server.
  - If you have not configured CMU for an on-premise database, you need to create these files. Then you upload the configuration files to the cloud to configure CMU on your Autonomous Database instance. You can validate the wallet and the `dsi.ora` by configuring CMU for an on-premise database and verifying that an Active Directory user can successfully log on to the on-premise database with these configuration files. Then you upload these configuration files to the cloud in order to configure CMU for your Autonomous Database.

  For details on the wallet file for CMU, see Create the Wallet for a Secure Connection and Verify the Oracle Wallet.

  For details on the `dsi.ora` file for CMU, see Creating the dsi.ora File.

  For details on configuring Active Directory for CMU and troubleshooting CMU for on-premise databases, see How To Configure Centrally Managed Users For Database Release 18c or Later Releases (Doc ID 2462012.1).

## Configure CMU with Microsoft Active Directory on Autonomous Database

To configure Autonomous Database for CMU to connect to Active Directory servers:

1. Connect to the Autonomous 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 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 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 Database for instructions to disable the access from Autonomous Database to Active Directory.

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

# Configure CMU with Microsoft Active Directory on Exadata Cloud@Customer

APPLIES TO: Exadata Cloud@Customer only

To configure Autonomous Database on Exadata Cloud@Customer for CMU to connect to Active Directory servers, without using Oracle Object Store service:

1. Connect to the Autonomous 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 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 Database directory object as demonstrated below:
   a. Create a Database directory in your Autonomous 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 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 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 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 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 Database on Exadata Cloud@Customer.

## Add Microsoft Active Directory Roles on Autonomous Database

To add Active Directory roles, map the database global roles to Active Directory groups with `CREATE ROLE` or `ALTER ROLE` statements (and include the `IDENTIFIED GLOBALLY AS` clause).

To add global roles for Active Directory groups on Autonomous Database:

1. Log in as the ADMIN user to the database that is configured to use Active Directory (the ADMIN user has the `CREATE ROLE` and `ALTER ROLE` system privileges that you need for these steps).
2. Set the database authorization for Autonomous Database roles with `CREATE ROLE` or `ALTER ROLE` statement. Include the `IDENTIFIED GLOBALLY AS` clause and specify the DN of an Active Directory group.

Use the following syntax to map a directory user group to a database global role:

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

For example:

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS
     'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com';
```

In this example all members of the `widget_sales_group` are authorized with the database role `widget_sales_role` when they log in to the database.

3. Use `GRANT` statements to grant the required privileges or other roles to the global role.

   For example:

   ```
   GRANT CREATE SESSION TO WIDGET_SALES_ROLE;
   GRANT DWROLE TO WIDGET_SALES_ROLE;
   ```

   `DWROLE` is a predefined role that has common privileges defined. See Manage Database User Privileges for information on setting common privileges for Autonomous Database users.

4. If you want to make an existing database role to be associated with an Active Directory group, then use `ALTER ROLE` statement to alter the existing database role to map the role to an Active Directory group.

   Use the following syntax to alter an existing database role to map it to an Active Directory group:

   ```
   ALTER ROLE existing_database_role
      IDENTIFIED GLOBALLY AS
   'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
   ```

5. If you want to create additional global role mappings for other Active Directory groups, follow these steps for each Active Directory group.

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

## Add Microsoft Active Directory Users on Autonomous Database

To add Active Directory users to access an Autonomous Database, map database global users to Active Directory groups or users with `CREATE USER` or `ALTER USER` statements (with `IDENTIFIED GLOBALLY AS` clause).

The integration of Autonomous Database with Active Directory works by mapping Microsoft Active Directory users and groups directly to Oracle database global users and global roles.

To add global users for Active Directory groups or users on Autonomous Database:

1. Log in as the ADMIN user to the database that is configured to use Active Directory (the ADMIN user has the required `CREATE USER` and `ALTER USER` system privileges that you need for these steps).

2. Set database authorization for Autonomous Database users with `CREATE USER` or `ALTER USER` statements and include the `IDENTIFIED GLOBALLY AS` clause, specifying the DN of an Active Directory user or group.

   Use the following syntax to map a directory user to a database global user:

   ```
   CREATE USER global_user IDENTIFIED GLOBALLY AS 'DN_of_an_AD_USER';
   ```

   Use the following syntax to map a directory group to a database global user:

   ```
   CREATE USER global_user IDENTIFIED GLOBALLY AS
       'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
   ```

   For example, to map a directory group named `widget_sales_group` in the `sales` organization unit of the `production.example.com` domain to a shared database global user named `WIDGET_SALES`:

   ```
   CREATE USER widget_sales IDENTIFIED GLOBALLY AS

   'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com';
   ```

   This creates a shared global user mapping. The mapping, with global user `widget_sales`, is effective for all users in the Active Directory group. Thus, anyone in the `widget_sales_group` can log in to the database using their Active Directory credentials (through the shared mapping of the `widget_sales` global user).

3. If you want Active Directory users to use an existing database user, own its schema, and own its existing data, then use `ALTER USER` to alter an existing database user to map the user to an Active Directory group or user.

   - Use the following syntax to alter an existing database user to map it to an Active Directory user:

     ```
     ALTER USER existing_database_user IDENTIFIED GLOBALLY AS
     'DN_of_an_AD_USER';
     ```

   - Use the following syntax to alter an existing database user to map it to an Active Directory group:

     ```
     ALTER USER existing_database_user
          IDENTIFIED GLOBALLY AS
     'DN_of_an_AD_GROUP_of_WHICH_the_AD_USER_IS_a_MEMBER';
     ```

4. If you want to create additional global user mappings for other Active Directory groups or users, follow these steps for each Active Directory group or user.

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

## Connect to Autonomous Database with Active Directory User Credentials

After the ADMIN user completes the CMU Active Directory configuration steps and creates global roles and global users, users log in to the Autonomous Database using their Active Directory username and password.

> **Note:**
>
> Do not log in using a Global User name. Global User names do not have a password and connecting with a Global User name will not be successful. You must have a global user mapping in your Autonomous Database in order to log in to the database. You cannot log in to the database with only global role mappings.

- To log in to the Autonomous Database using an Active Directory username and password, connect as follows:

  ```
  CONNECT "AD_DOMAIN\AD_USERNAME"/
  AD_USER_PASSWORD@TNS_ALIAS_OF_THE_AUTONOMOUS_DATABASE;
  ```

  For example:

  ```
  CONNECT "production\pfitch"/password@adbname_medium;
  ```

  You need to include double quotes when the Active Directory domain is included along with the username, as with this example: `"production\pfitch"`.

  In this example, the Active Directory username is `pfitch` in domain `production`. The Active Directory user is a member of `widget_sales_group` group which is identified by its DN `'CN=widget_sales_group,OU=sales,DC=production,DC=example,DC=com'`.

After configuring CMU with Active Directory on Autonomous Database and setting up Active Directory authorization, with global roles and global users, you can connect to your Autonomous Database using any of the connection methods described in About Connecting to a Dedicated Autonomous Database . When you connect, if you want to use an Active Directory user then use Active Directory user credentials. For example, provide a username in this form, "*AD_DOMAIN\AD_USERNAME*" (double quotes must be included), and use your *AD_USER_PASSWORD* for the password.

## Verify Active Directory User Connection Information with Autonomous Database

When users log in to the Autonomous 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 Database global user.

After `pfitch` logs in to the database, the command `SHOW USER` shows the global user name:

```
SHOW USER;

USER is "WIDGET_SALES"
```

The following command shows the DN (Distinguished Name) of the Active Directory user:

```
SELECT SYS_CONTEXT('USERENV', 'ENTERPRISE_IDENTITY') FROM DUAL;
```

For example you can verify this centrally managed user's enterprise identity:

```
SQL> SELECT SYS_CONTEXT('USERENV', 'ENTERPRISE_IDENTITY') FROM DUAL;

SYS_CONTEXT('USERENV','ENTERPRISE_IDENTITY')
--------------------------------------------------------------------------------
cn=Peter Fitch,ou=sales,dc=production,dc=examplecorp,dc=com
```

The following command shows the "`AD_DOMAIN\AD_USERNAME`":

```
SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATED_IDENTITY') FROM DUAL;
```

For example, the Active Directory authenticated user identity is captured and audited when the user logs on to the database:

```
SQL> SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATED_IDENTITY') FROM DUAL;

SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY')
--------------------------------------------------------------------------------
production\pfitch
```

See Verifying the Centrally Managed User Logon Information for more information.

## Remove Active Directory Users and Roles on Autonomous Database

To remove Active Directory users and roles from Autonomous Databases, use standard database commands. This does not remove the related Active Directory users or groups that were mapped from the dropped database users or roles.

To remove users or roles from Autonomous Database:

1. Log in to the database that is configured to use Active Directory as a user who has been granted the `DROP USER` or `DROP ROLE` system privilege.
2. Drop the global users or the global roles that are mapped to Active Directory groups or users with `DROP USER` or `DROP ROLE` statement.

   See Remove Database Users for more information.

## Disable Active Directory Access on Autonomous Database

Describes the steps to remove the CMU configuration from your Autonomous Database (and disable the LDAP access from your Autonomous Database to Active Directory).

After you configure your Autonomous Database instance to access CMU Active Directory, you can disable the access as follows:

1. Connect to the Autonomous Database as the ADMIN user.
2. Use the `DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION` to disable CMU authentication.

> **Note:**
>
> To run this procedure you must be logged in as ADMIN user or have the `EXECUTE` privilege on `DBMS_CLOUD_ADMIN`.

For example:

```
BEGIN
   DBMS_CLOUD_ADMIN.DISABLE_EXTERNAL_AUTHENTICATION;
END;
/
```

This disables CMU authentication on your Autonomous Database instance.

See DISABLE_EXTERNAL_AUTHENTICATION Procedure for more information.

## Limitations with Microsoft Active Directory on Autonomous Database

The following limitations apply to CMU with Active Directory on Autonomous Database:

- Only "password authentication" and Kerberos is supported for CMU with Autonomous Database. When you are using CMU authentication with Autonomous 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 Database.

# Develop

## Developing Applications with Autonomous 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 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 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 Databases for further reference.

Autonomous Database for Developers is a free Autonomous Database that developers can use to build and test new applications. It supports all the developer features and tools provided by Oracle Autonomous 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 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 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 Database:

- **The Oracle Cloud console.** For instructions, see Create a Dedicated Autonomous 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 for more information.

**Notes:**

- `CREATE DIRECTORY` creates the database directory object in the Autonomous 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 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 for more information.

**Notes:**

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

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

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

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

  To run `DBMS_CLOUD.LIST_FILES` with a user other than ADMIN you need to grant read privileges on the directory to that user. See LIST_FILES Function for more information.
- The `DROP DIRECTORY` command does not remove the underlying file system directory. The Autonomous 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 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 Database Dedicated 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 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 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 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 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 Database Deployments to learn about using Oracle Database features and options in Autonomous 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 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 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.

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

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

1. Download and install the Oracle Instant Client basic package for your system's OS and architecture:
   - **Oracle Linux:**
     a. Enable the Instant Client channel:

        ```
        yum -y install oracle-release-el7
        ```

     b. Download and install the Instant Client 19.3 basic package:

        ```
        yum -y install oracle-instantclient19.3-basic
        ```

   - **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. See Download Client Credentials for information on downloading client credentials for Autonomous Database.
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 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 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 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 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 Database

## Build a Java Application

To build a Java application that accesses an Autonomous 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 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 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 Database, perform these steps.

1. Download and install the Java Development Kit (JDK).
2. Download the client credentials for your Autonomous 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 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 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 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 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 Database.

   - `ojdbc8.jar`: the core JDBC driver
   - `oraclepki.jar`, `osdt_core.jar`, and `osdt_cert.jar`: for an Autonomous 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 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 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 *Oracle Database JDBC Developer's Guide* and *Oracle Database JDBC Java API Reference*.

For detailed information about the Universal Connection Pool, see *Oracle Universal Connection Pool Developer's Guide* and *Oracle Universal Connection Pool API Reference*.

## Build a .NET Application with Visual Studio and Visual Studio Code

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

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

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

Oracle Developer Tools for Visual Studio provides database application design-time support in Visual Studio, including tools for managing Autonomous 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 Database.

### Set-Up Instructions

Refer to Developing .NET Applications for Oracle Autonomous Database for instructions on how to download, install, and configure these components for use with Oracle Autonomous Database.

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

### 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>
      <settings>
        <setting name="WALLET_LOCATION" value="<wallet-directory>" />
  ```

```
    </settings>
  </version>
</oracle.unmanageddataaccess.client>
```

## Build a Node.js Application

To build a Node.js application that accesses an Autonomous 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 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 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 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 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 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 Node.js application connectivity to an Autonomous 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 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 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 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 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 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 = On > /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 = On
     ```

### 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 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 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 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 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 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 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 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 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 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 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 Python application connectivity to an Autonomous 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 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 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 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 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 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 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 Ruby application connectivity to an Autonomous 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 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 Database on Dedicated Exadata Infrastructure supports application development with Oracle Application Express (APEX). Oracle APEX on Autonomous 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 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 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 Database Dedicated for Developers and Database Users Workshop.

### Access Oracle Application Express in Autonomous Database

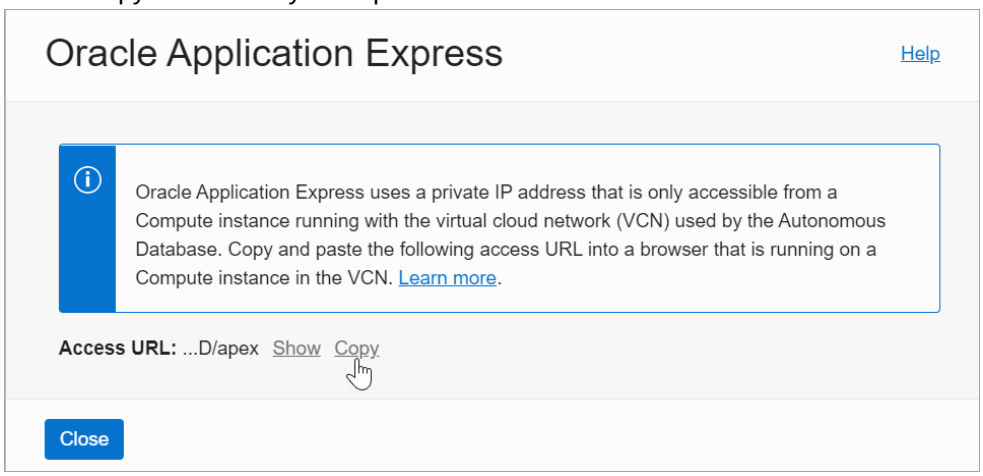
> **Note:**
>
> To access Oracle Applications Express, the system you are using must have network access to the Autonomous Database, as described in Configure a Development System to Access the Database.

Learn how to access Oracle Application Express from Autonomous 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 Database. See View Details of a Dedicated Autonomous 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 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 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 Database, as described in Configure a Development System to Access the Database.

Learn how to access Oracle Application Express administration services from an Autonomous 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 Database **Details** page, and paste it in your web browser's address bar. For more details, see Access Oracle Application Express in Autonomous 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 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 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 Database automatically upgrades your Oracle Application Express instance when you schedule an upgrade through Service Request (SR) in My Oracle Support. You must supply the OCIDs of the tenancy and Autonomous 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 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 Database

You can develop and deploy RESTful Services with native Oracle REST Data Services (ORDS) support on an Autonomous Database.

## About Oracle REST Data Services in Autonomous Database

Oracle REST Data Services (ORDS) makes it easy to develop REST interfaces for relational data in an Autonomous Database. ORDS is a mid-tier Java application that maps HTTP(S) verbs, such as GET, POST, PUT, DELETE, and so on, to database transactions, and returns any results as JSON data.

> **Note:**
>
> The Oracle REST Data Services (ORDS) application in Autonomous Database is preconfigured and fully managed. ORDS connects to the database using the `low` predefined database service with a fixed maximum number of connections (the number of connections for ORDS does not change based on the number of 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 Databases for information on the `low` database service.

## Develop with Oracle REST Data Services on Autonomous Database

Autonomous Database supports Oracle REST Data Services (ORDS).

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

- SQL Developer (desktop): With SQL Developer on your desktop, you can connect to your Autonomous 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 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 Database

## Send Email on Autonomous Database on Dedicated Exadata Infrastructure

APPLIES TO: ✔ Oracle Public Cloud only

You can send email on Autonomous 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 Database

Describes the steps to send email using `UTL_SMTP` on Autonomous 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 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 for further reference.

See Notes on the Use of Database PL/SQL Packages for more information about using `UTL_SMTP` with Autonomous Database.

## SMTP Send Email Sample Code

Shows sample code for sending email with `UTL_SMTP` on an Autonomous 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 Database

You can use the `DBMS_CLOUD_NOTIFICATION` to send messages as an email.

> **Note:**
>
> You can only use `DBMS_CLOUD_NOTIFICATION` for Email notifications with Autonomous Database version 19.21 and above.

1. Identify your SMTP connection endpoint for Email Delivery. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region.

   For example, select one of the following for the SMTP connection endpoint:

   - smtp.us-phoenix-1.oraclecloud.com
   - smtp.us-ashburn-1.oraclecloud.com
   - smtp.email.uk-london-1.oci.oraclecloud.com
   - smtp.email.eu-frankfurt-1.oci.oraclecloud.com

   See Configure SMTP Connection for more information.

2. Generate SMTP credentials for Email Delivery. `UTL_SMTP` uses credentials to authenticate with Email Delivery servers when you send email.

   See Generate SMTP Credentials for a User for more information.

3. Create an approved sender for Email Delivery. Complete this step for all email addresses you use as the "`From`" with `UTL_SMTP.MAIL`.

   See Managing Approved Senders for more information.

4. Create a credential object and use `DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE` to send a message as an email.

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'EMAIL_CRED',
        username        => 'username',
        password        => 'password'
   );
   END;
/
BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider        => 'email',
        credential_name => 'EMAIL_CRED',
        message         => 'Subject content',
        params          => json_object('recipient' value
'mark@example.com, suresh@example.com',
                                       'to_cc'  value
'nicole@example.com, jordan@example.com',
                                       'to_bcc' value
'manisha@example.com',
                                       'subject' value 'Test
subject',
                                       'smtp_host' value
'smtp.email.example.com',
                                       'sender'    value
'approver_sender@example.com' )
   );
   END;
/
```

   Use the `params` parameter to specify the sender, smtp_host, subject, recipient, and recipients of a CC or BCC in `string` values.

   - *sender*: specifies the Email ID of the approved sender from Step 3.
   - *smtp_host*: specifies the SMTP host name from step 2.
   - *subject*: specifies the subject of the email.
   - *recipient*: specifies the email IDs of recipients. Use a comma between email IDs when there are multiple recipients.
   - *to_cc*: specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients.
   - *to_bcc*: specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients.

   See SEND_MESSAGE Procedure for more information.

## Send Query Results as Email from Autonomous Database

You can use the `DBMS_CLOUD_NOTIFICATION` package to send the results of a query as an email.

> **Note:**
>
> You can only use `DBMS_CLOUD_NOTIFICATION` for Email notifications with Autonomous Database version 19.21 and above.

1. Identify your SMTP connection endpoint for Email Delivery. You may need to subscribe to additional Oracle Cloud Infrastructure regions if Email Delivery is not available in your current region.

   For example, select one of the following for the SMTP connection endpoint:

   - smtp.us-phoenix-1.oraclecloud.com
   - smtp.us-ashburn-1.oraclecloud.com
   - smtp.email.uk-london-1.oci.oraclecloud.com
   - smtp.email.eu-frankfurt-1.oci.oraclecloud.com

   See Configure SMTP Connection for more information.

2. Generate SMTP credentials for Email Delivery. `UTL_SMTP` uses credentials to authenticate with Email Delivery servers when you send email.

   See Generate SMTP Credentials for a User for more information.

3. Create an approved sender for Email Delivery. Complete this step for all email addresses you use as the "`From`" with `UTL_SMTP.MAIL`.

   See Managing Approved Senders for more information.

4. Create a credential object and use `DBMS_CLOUD_NOTIFICATION.SEND_DATA` to send the output of a query as an email.

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
        credential_name => 'EMAIL_CRED',
        username        => 'username',
        password        => 'password'
   );
   END;
/
BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_DATA(
        provider        => 'email',
        credential_name => 'EMAIL_CRED',
        query           => 'SELECT tablespace_name FROM dba_tablespaces',
        params          => json_object('recipient' value
'mark@example.com, suresh@example.com',
                                       'to_cc'  value
'nicole@example.com1, jordan@example.com',
                                       'to_bcc' value
```

```
'manisha@example.com',
                                          'subject' value 'Test
subject',
                                          'type' value 'json',
                                          'title' value 'mytitle',
                                          'message' value 'This is the
message',
                                          'smtp_host' value
'smtp.email.example.com',
                                          'sender'    value
'approver_sender@example.com' )
   );
   END;
/
```

Use the `params` parameter to specify the sender, smtp_host, subject, recipient, recipients of a CC or BCC, the message, data type, and the title in `String` values.

- *sender*: specifies the Email ID of the approved sender from Step 3.
- *smtp_host*: specifies the SMTP host name from step 2.
- *subject*: specifies the subject of the email. The maximum size is 100 characters.
- *recipient*: This specifies the email IDs of recipients. Use a comma between email IDs when there are multiple recipients.
- *to_cc*: specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients.
- *to_bcc*: specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients.
- *message*: specifies the message text.
- *type*: specifies the output format as either CSV or JSON.
- *title*: specifies the title of the attachment of SQL output. The title should only contain letters, digits, underscores, hyphens, or dots as characters in its value due to it being used to generate a file name.

See SEND_DATA Procedure for more information.

## Send Microsoft Teams Notifications from Autonomous Database on Dedicated Exadata Infrastructure

You can configure Microsoft Teams to send messages, alerts, or output of a query from Autonomous Database on Dedicated Exadata Infrastructure to a Microsoft Teams channel using the `DBMS_CLOUD_NOTIFICATION` package.

### Prepare to Send Microsoft Teams Notifications from Autonomous Database

Get started by configuring a bot in your Microsoft Teams app. Next, create a credential to use with the `DBMS_CLOUD_NOTIFICATION` procedures to send Microsoft Teams notifications from Autonomous Database.

To configure Microsoft Teams notifications:

1. Create your Microsoft Teams app and add a bot in it. See Developer Portal for Teams for information on setting up an app.
2. From the **Bot Management** section, ensure that the bot has a secret key, scope set to *Team* and permission to send notifications.
3. Publish the app to your org to make it available to the people in your org.
4. After your IT admin approves the app from the admin center, install the app from the **Apps** section in Teams.
5. Request the `Files.ReadWrite.All` and `ChannelSettings.Read.All` permissions to the app for Graph API from the Azure Portal using the following instructions:
   a. Log into your Azure Portal, navigate to **Azure Active Directory** using the left panel, and select the **Application** option.
   b. The Application page displays the apps that you own along with the bots added to those apps. Click the bot to view its details.
   c. Copy the *directory/tenant id* from the bot overview page for later use.
   d. Then, go to **API permissions** in the left panel. Under **API permissions**, click **Add permission**, select **Microsoft graph** and then **Application permission**.
   e. Search for `Files.ReadWrite.All` and `ChannelSettings.Read.All` permissions and add them.
6. Have your IT admin approve the above-requested permissions from the Azure portal by following the steps below:
   a. Log into your Azure Portal, navigate to **Azure Active Directory** using the left panel, and select the **Application** option.
   b. Select **All applications** from the **Application** page.
   c. Search the *application/bot* by its name, go to **API permissions**, and grant the ADMIN consent for the requested permissions: `Files.ReadWrite.All` and `ChannelSettings.Read.All`.

> **Tip:**
>
> After your app is approved by the IT Admin, you can provide the bot id and secret key to other users to install the app within Teams in the org.

7. After the app is approved by the IT admin and the above requested permissions are granted, you can use the app's bot ID and secret key are used to create the credential object and generate a bot token.
8. To send a query result to a Microsoft Teams channel, obtain the `team id` and the `tenant id`.

> **Tip:**
>
> The `team id` is located in the team link between /team/ and /conversations. The `tenant id` is located after "tenantId=" at the end of the team link. This link is found by clicking the three dots next to the team name and selecting **Get link to team**.

For example:

```
https://teams.microsoft.com/l/team/teamID/conversations?
groupId=groupid%tenantId=tenantid
```

9. Obtain the `channelID`.

> **Tip:**
>
> `channelID` is located in the channel link between /team/ and the channel name. This link is found by clicking the three dots next to the channel name and selecting **Get link to channel**.

For example:

```
https://teams.microsoft.com/l/channel/channelID/channel_name?
groupId=groupid&tenantId=tenantid
```

10. 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 Database.

> **Tip:**
>
> If you can not use the `CREATE_CREDENTIAL` procedure successfully, consult with the ADMIN user to grant execute access on `DBMS_CLOUD` packages.

The credential's username is the bot_id and the password is the bot key.

For example:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(credential_name =>
'TEAMS_CRED',
        username        => 'bot_id',
```

```
        password        => 'bot_secret');
END;
/
```

See CREATE_CREDENTIAL Procedure for more information.

## Send Messages to a Microsoft Teams Channel

After creating the Microsoft Teams credential object as described in Prepare to Send Microsoft Teams Notifications from Autonomous Database, you can use the `DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE` procedure to send a message to a Microsoft Teams channel.
**Example:**

```
BEGIN
     DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider        => 'msteams',
        credential_name => 'TEAMS_CRED',
        message         => 'text from new teams api',
        params          => json_object('channel' value 'channelID'));
END;
/
```

Use the `params` parameter to specify the channel.

- *channel*: specifies the Channel ID obtained from Step 10 in Prepare to Send Microsoft Teams Notifications from Autonomous Database.

See SEND_MESSAGE Procedure for more information.

## Send Query Results to a Microsoft Teams Channel

After creating the Microsoft Teams credential object as described in Prepare to Send Microsoft Teams Notifications from Autonomous Database, you can use the `DBMS_CLOUD_NOTIFICATION.SEND_DATA` procedure to send the output of a query to a Microsoft Teams channel.
**Example:**

```
BEGIN
     DBMS_CLOUD_NOTIFICATION.SEND_DATA(provider => 'msteams',
        credential_name => 'TEAMS_CRED',
        query           => 'SELECT tablespace_name FROM dba_tablespaces',
        params          => json_object('tenant'value '5b743bc******c0286',
                                       'team'value '0ae401*********5d2bd',
                                       'channel'value
'19%3a94be023*****%40thread.tacv2',
                                       'title'value 'today',
                                       'type'value 'csv'));
END;
/
```

Use the `params` parameter to specify the tenant, team, channel, title, and data type in `string` values.

- *tenant*: specifies the tenant ID obtained from Step 8 in Prepare to Send Microsoft Teams Notifications from Autonomous Database.
- *team*: specifies the team ID obtained from Step 8 in Prepare to Send Microsoft Teams Notifications from Autonomous Database.
- *channel*: specifies the channel ID obtained from Step 9 in Prepare to Send Microsoft Teams Notifications from Autonomous Database.
- *title*: specifies the title of the file. The title can only contain alphabets, digits, underscores, and hyphens. The file name that appears in Microsoft Teams will be a concatenation of the title parameter and the timestamp to ensure uniqueness. The maximum title size is 50 characters.

  For example: `'title'_'timestamp'.'format'`
- *type*: This specifies the output format. Valid values are CSV or JSON.

> **Note:**
>
> The maximum file size supported when using `DBMS_CLOUD_NOTIFICATION.SEND_DATA` for Microsoft Teams is 4MB.

See SEND_DATA Procedure for more information.

## Send Slack Notifications from Autonomous Database on Dedicated Exadata Infrastructure

You can configure Slack to send messages, alerts, or output of a query from Autonomous Database on Dedicated Exadata Infrastructure to a Slack channel using the `DBMS_CLOUD_NOTIFICATION` package.

### Prepare to Send Slack Notifications from Autonomous Database

To send Slack notifications, you must configure your Slack application to receive messages from Autonomous Database. Next, create a credential to use with the `DBMS_CLOUD_NOTIFICATION` procedures to send Slack notifications from Autonomous Database.

To use Slack with `DBMS_CLOUD_NOTIFICATION` procedures:

1. Create your Slack app.

   The Slack app is installed in a Slack Workspace, which in turn has Channels where messages can be sent. The bot token of the Slack app must have the following permission scopes defined:

   ```
   channels:read
   chat:write
   files:write
   ```

   See Creating an app for information on setting up a Slack app.

2. Have your Slack admin add the Slack app to the channels to which `DBMS_CLOUD_NOTIFICATION` can send message through the "Integrations" option in the channel.

   See Basic app setup for more information.

3. Locate the app's bot token available in the Slack app specific page, under `https://app.slack.com`.

   See Basic app setup for more information.

4. 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 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;
```

See Configuring Access Control for External Network Services for more information.

## Send Messages to a Slack Channel

After creating the Slack credential object as described in Prepare to Send Slack Notifications from Autonomous Database, you can use the `DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE` procedure to send a message to a Slack channel.

**Example:**

```
BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
      provider          => 'slack',
      credential_name   => 'SLACK_CRED',
      message           => 'Alert from Autonomous Database...',
      params            => json_object('channel' value 'C0....08'));
END;
/
```

Use the `params` parameter to specify the Slack channel.

- `channel`: Specifies the Channel ID.

  The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab. See How to Find your Slack Team ID and Slack Channel ID for more information.

See SEND_MESSAGE Procedure for more information.

## Send Query Results to a Slack Channel

After creating the Slack credential object as described in Prepare to Send Slack Notifications from Autonomous Database, you can use the `DBMS_CLOUD_NOTIFICATION.SEND_DATA` procedure to send the output of a query to a Slack channel.

**Example:**

```
BEGIN
  DBMS_CLOUD_NOTIFICATION.SEND_DATA(
      provider => 'slack',
      credential_name => 'SLACK_CRED',
      query => 'SELECT username, account_status, expiry_date FROM
USER_USERS WHERE rownum < 5',
      params => json_object('channel' value 'C0....08',
                            'type' value 'csv'));
```

```
END;
/
```

Use the `params` parameter to specify the Slack channel and the data type:

- `channel`: Specifies the Channel ID.

  The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab. See How to Find your Slack Team ID and Slack Channel ID for more information.

- `type`: Specifies the output type. Valid values are: '`csv`' or '`json`'.

See SEND_DATA Procedure for more information.

# Manage Data

## About Cloning Autonomous Database on Dedicated Exadata Infrastructure

Cloning is the process of creating a point-in-time copy of your Autonomous Database or its backup set. You can use the cloning feature to quickly set up an Autonomous 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 Database on Dedicated Exadata Infrastructure) after the clone operation completes.

### Clone Types

Autonomous 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 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.

2. **A backup of a database instance:** You can create a new database instance by cloning an automatic backup of the Autonomous 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.

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.
- 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 Database instance or its backup set successfully, the following requirements have to be met:

- To clone an Autonomous 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.
- 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 **1 OCPU** or **4 ECPUs** 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:
    * On Exadata Cloud@Customer, if both the source and target use customer managed keys using Oracle Key Vault (OKV), they must use the same OKV destination and 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.

## Clone Limitations

There are a few limitations with cloning Autonomous 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.
- 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 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 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 Database depends on the CPU Count and the network bandwidth between the Backup Destination and the target Autonomous Container Database.

## Step-by-Step Guides

To learn how to clone an Autonomous Database from different clone sources using the OCI console, see:

- Clone an Autonomous Database Instance
- Clone Autonomous 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.

# Integrate

## Analyzing and Visualizing Data

Gain insight into the data in your Autonomous 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 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 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 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 Database follow *Lab 24: Oracle Analytics Cloud* in Oracle Autonomous Database Dedicated 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 Database

To learn about and use Oracle Analytics Cloud, visit the Oracle Analytics Cloud home page.

## Querying External Data with Autonomous Database on Dedicated Exadata Infrastructure

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

> **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 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 `rejectlimit` to suppress these errors. As an alternative, you can also validate the external table you created to see the error messages and the rejected rows so that you can change your format options accordingly. See Validate External Data for more information.

For detailed information about the parameters, see CREATE_EXTERNAL_TABLE Procedure.

## Validate External Data

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

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

```
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.

# Oracle Extensions for IDEs

Oracle extensions let developers connect to, browse, and manage Autonomous Databases directly from common IDEs.

## Use Oracle Cloud Infrastructure Toolkit for Eclipse

Oracle Cloud Infrastructure Toolkit for Eclipse is a plugin that enables Java developers to easily connect to Oracle Autonomous Database 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 Databases, stopping and starting, scaling up and down, and so on. You can also use the plugin to easily connect to the databases to browse the schema, access tables, execute SQL statements, and perform other development tasks.

Users with permissions to manage the databases can perform a number of actions, including those listed below. For detailed information about permissions, see Toolkit for Eclipse in the Oracle Cloud Infrastructure documentation. You can:

- Create Autonomous Databases
- Start, stop, terminate, clone, and restore Autonomous Databases
- Scale up and down
- Download the client credentials zip file (database wallet)
- Connect to Autonomous Databases
- Browse the schema
- Choose compartments and regions
- Change the administrator password, update the license type, and so on

Download the latest version of the plugin from GitHub (`com.oracle.oci.eclipse-`*`version`*`.zip`, where *`version`* is the latest version, for instance `1.2.0`):

`https://github.com/oracle/oci-toolkit-eclipse/releases`

Then follow the installation instructions and details about how to get started in this step-by-step walkthrough:

New Eclipse Plugin for Accessing Autonomous Database (ATP/ADW)

## Use Oracle Developer Tools for Visual Studio

Oracle Developer Tools for Visual Studio is a tightly integrated extension for Microsoft Visual Studio and Oracle Autonomous Database 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 Databases, stopping and starting, scaling up and down, and so on. You can also use the extension to easily connect to the databases and perform development tasks, such as browsing your Oracle schema and launching integrated Oracle designers and wizards to create and alter schema objects.

Users with permissions to manage the databases can perform a number of actions, including the following:

- Sign up for Oracle Cloud
- Connect to a cloud account using a simple auto-generated config file and key file
- Create new or clone existing Always Free Autonomous Database, Autonomous Database Dedicated, and Autonomous Database Serverless databases
- Automatically download credentials files (including wallets) and quickly connect, browse, and operate on Autonomous Database schemas
- Change compartments and regions without reconnecting
- Start, stop, or terminate Autonomous Database
- Scale up/down Autonomous Database resources
- Restore from backup
- Update instance credentials, update the license type used
- Rotate wallets
- Convert Always Free Autonomous Database into paid databases

> **Note:**
>
> Promotion of Always Free to paid is currently supported *only* if the Always Free instance has database release *19c*.

Download the extension from Visual Studio Marketplace:

- Oracle Developer Tools for Visual Studio 2019
- Oracle Developer Tools for Visual Studio 2017

You'll find lots of information about the extension on those Marketplace pages.

Then follow the installation instructions and details about how to get started in this step-by-step walkthrough:

New Release: Visual Studio Integration with Oracle Autonomous Database

For detailed information about how to use the extension, see the online documentation that's optionally installed with Oracle Developer Tools for Visual Studio. Press the F1 key to display the context-sensitive help for each dialog.

## Use Oracle Developer Tools for VS Code

Oracle Developer Tools for VS Code is a tightly integrated extension for Microsoft Visual Studio Code (VS Code) and Oracle Autonomous Database 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 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 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 Database, you must follow a sequence of steps to use OCMT. They are:

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

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

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

# Use Select AI to Generate SQL from Natural Language Prompts

Autonomous Database on Dedicated Exadata Infrastructure Select AI enables you to query your data using natural language.
The Select AI feature allows Autonomous Database to use generative AI with Large Language Models (LLMs) to convert user's input text into Oracle SQL. Select AI processes the natural language prompt, supplements the prompt with metadata, and then generates and runs a SQL query.

## Usage Guidelines

Provides usage guidelines that ensure effective and proper usage of natural language prompts for SQL generation to ensure an enhanced user experience.

**Intended Use**

This feature is intended for the generation and running of SQL queries resulting from user-provided natural language prompts. It automates what a user could do manually based on their schema metadata in combination with a large language model (LLM) of their choice.

While any prompt can be provided, including those that do not relate to the production of SQL query results, Select AI focuses on SQL query generation. Select AI enables submitting general requests with the `chat` action.

**Prompt Augmentation Data**

The database augments the user-specified prompt with database metadata to mitigate hallucinations from the LLM. The augmented prompt is then sent to the user-specified LLM to produce the query.

The database augments the prompt with schema metadata only. This metadata may include schema definitions, table and column comments, and content available from the data dictionary and catalog. For the purposes of SQL generation, the database does not provide table or view contents (actual row or column values) when augmenting the prompt.

The `narrate` action, however, does provide the result of the query, which may contain database data, to the user-specified LLM from which to generate natural language text describing the query results.

> **WARNING:**
>
> Large language models (LLMs) have been trained on a broad set of text documentation and content, typically from the Internet. As a result, LLMs may have incorporated patterns from invalid or malicious content, including SQL injection. Thus, while LLMs are adept at generating useful and relevant content, they also can generate incorrect and false information including SQL queries that produce inaccurate results and/or compromise security of your data.
>
> The queries generated on your behalf by the user-specified LLM provider will be run in your database. Your use of this feature is solely at your own risk, and, notwithstanding any other terms and conditions related to the services provided by Oracle, constitutes your acceptance of that risk and express exclusion of Oracle's responsibility or liability for any damages resulting from that use.

## About SQL Generation

Using natural language to interact with your database data is now achievable with LLMs. This means you can use natural language, for example plain English, to query the database.

When you use Select AI, Autonomous Database manages the process of converting natural language into SQL. This means you can provide a natural language prompt instead of SQL code to interact with your data. Select AI serves as a productivity tool for SQL users and developers and enables non-expert SQL users to derive useful insights from their data, without having to understand data structures or technical languages.

The `DBMS_CLOUD_AI` package in Autonomous Database enables integration with a user-specified LLM for generating SQL code using natural language prompts. The package assists in supplying the LLM with knowledge of the database schema and instructing it to write a SQL query consistent with that schema. The `DBMS_CLOUD_AI` package works with AI providers like OpenAI, Cohere, Azure OpenAI Service, and Oracle Cloud Infrastructure Generative AI.

> **Note:**
>
> Users must have an account with the AI provider and provide their credentials through `DBMS_CLOUD_AI` objects that the Autonomous Database uses.

## Use DBMS_CLOUD_AI to Configure AI Profiles

Autonomous Database uses AI profiles to facilitate and configure access to an LLM and to setup for the generation of SQL statements from natural language prompts.

AI profiles include database objects that are the target for natural language queries. Metadata used from these targets can include database table names, column names, column data types, and comments. You create and configure AI profiles using the `DBMS_CLOUD_AI.CREATE_PROFILE` and `DBMS_CLOUD_AI.SET_PROFILE` procedures.

## Requirements to Configure DBMS_CLOUD_AI Package

The following are required to run `DBMS_CLOUD_AI`:

- Access to an Oracle Cloud Infrastructure cloud account and to an Autonomous Database instance.
- A paid API account for a supported AI provider, one of:
  - **OpenAI:** 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 Use AI Keyword to Enter Prompts.
- Network ACL privileges to access your external AI provider.
- A credential that provides access to the AI provider.

## Configure DBMS_CLOUD_AI Package

Describes the steps to use `DBMS_CLOUD_AI`.

**Configure `DBMS_CLOUD_AI`**

To configure `DBMS_CLOUD_AI`:

1. Grant the `EXECUTE` privilege on the `DBMS_CLOUD_AI` package to the user who wants to use Select AI.

By default, only ADMIN user is granted the `EXECUTE` privilege. The ADMIN user can grant `EXECUTE` privilege to other users.

2. Grant network ACL access to the user who wants to use Select AI and for the AI provider endpoint.

   The ADMIN user can grant network ACL access. See APPEND_HOST_ACE Procedure for more information.

3. Create a credential to enable access to your AI provider.
   See CREATE_CREDENTIAL Procedure for more information.

The following example grants the `EXECUTE` privilege to `ADB_USER`:

```
grant execute on DBMS_CLOUD_AI to ADB_USER;
```

The following example grants `ADB_USER` the privilege to use the *api.openai.com* endpoint.

```
BEGIN
    DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
         host => 'api.openai.com',
         ace  => xs$ace_type(privilege_list => xs$name_list('http'),
                             principal_name => 'ADB_USER',
                             principal_type => xs_acl.ptype_db)
   );
END;
/
```

### APPEND_HOST_ACE Function Parameters

| Parameter | Description |
|---|---|
| `host` | The host, which can be the name or the IP address of the host. You can use a wildcard to specify a domain or a IP subnet. The host or domain name is not case sensitive.<br>For OpenAI, use *api.openai.com*.<br>For Cohere, use *api.cohere.ai*.<br>For Azure OpenAI Service, use *<azure_resource_name>.openai.azure.com*. See Profile Attributes to know more about `azure_resource_name`. |
| `ace` | The access control entries (ACE). The `XS$ACE_TYPE` type is provided to construct each ACE entry for the ACL. For more details, see Creating ACLs and ACEs. |

Here is an example of how to create a credential to enable access to OpenAI.

```
EXEC DBMS_CLOUD.CREATE_CREDENTIAL('OPENAI_CRED', 'OPENAI', 'your_api_token');
```

### `DBMS_CLOUD.CREATE_CREDENTIAL` Parameters

| Parameter | Description |
|---|---|
| `credential_name` | The name of the credential to be stored. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens. |

| Parameter | Description |
|---|---|
| username | The username and password arguments together specify your AI provider credentials.<br>The username is a user-specified user name. |
| password | The username and password arguments together specify your AI provider credentials.<br>The password is your AI provider secret API key, and depends on the provider, that is, OpenAI, Cohere, or Azure OpenAI Service. See Requirements to Configure DBMS_CLOUD_AI Package 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 |
|---|---|
| `runsql` | Run the provided SQL command using a natural language prompt. This is the default action and it is optional to specify this parameter. |
| `showsql` | Displays the SQL statement for a natural language prompt. |
| `narrate` | The output of the prompt is explained in natural language. This option sends the SQL result to the AI provider to produce a natural language summary. |
| `chat` | Generates a response directly from the LLM based on the prompt. If `conversation` in the `DBMS_CLOUD_AI.CREATE_PROFILE` function is set to `true`, this option includes content from prior interactions or prompts, potentially including schema metadata. |
| `explainsql` | The SQL generated from the prompt is explained in natural language. This option sends the generated SQL to the AI provider to produce a natural language explanation. |

### Usage Notes

- Select `AI` is not supported in Database Actions or APEX Service. You can use only `DBMS_CLOUD_AI.GENERATE` function.
- The `AI` keyword is supported only in a `SELECT` statement.
- You cannot run PL/SQL statements, DDL statements, or DML statements using the `AI` keyword.

- The sequence is `SELECT` followed by `AI`. These keywords are not case-sensitive. After a `DBMS_CLOUD_AI.SET_PROFILE` is configured, the text after `SELECT AI` is a natural language prompt. If an AI profile is not set, `SELECT AI` reports the following error:

  ```
  ORA-00923: FROM keyword not found where expected
  00923. 00000 -  "FROM keyword not found where expected"
  ```

- Special character usage rules apply according to Oracle guidelines. For example, use single quotes twice if you are using an apostrophe in a sentence.

  ```
  select ai how many customers in SF don''t own their own home
  ```

- LLMs are subject to *hallucinations* and results are not always correct:
  - It is possible that `SELECT AI` may not be able to run the generated SQL for a specific natural language prompt.
  - It is possible that `SELECT AI` may not be able to generate SQL for a specific natural language prompt.

  In such a scenario, `SELECT AI` responds with information to assist you in generating valid SQL.
- Use the `chat` action, with `SELECT AI chat`, to learn more about SQL constructs. For better results with the `chat` action, use database views or tables with contextual column names or consider adding column comments explaining values stored in the columns.
- To access DBA or USER views, see DBMS_CLOUD_AI Views.

## Terminology

Definition of some of the terms used in Select AI feature are described.

The following are the terms related to Select AI feature:

| Term | Definition |
|---|---|
| Database Credential | Database Credentials are authentication credentials used to access and interact with databases. They typically consist of a user name and a password, sometimes supplemented by additional authentication factors like security tokens. These credentials are used to establish a secure connection between an application or user and a database, ensuring that only authorized individuals or systems can access and manipulate the data stored within the database. |
| Hallucination in LLM | Hallucination in the context of Large Language Models refers to a phenomenon where the model generates text that is incorrect, nonsensical, or unrelated to the input prompt. Despite being a result of the model's attempt to generate coherent text, these instances can contain information that is fabricated, misleading, or purely imaginative. Hallucination can occur due to biases in training data, lack of proper context understanding, or limitations in the model's training process. |

| Term | Definition |
| --- | --- |
| Large Language Model (LLM) | Large Language Models refer to advanced artificial intelligence models that are trained on massive amounts of text data to understand and generate human-like language, software code, and database queries. These models are capable of performing a wide range of natural language processing tasks, including text generation, translation, summarization, question answering, sentiment analysis, and more. LLMs are typically neural network-based architectures that learn patterns, context, and semantics from the input data, enabling them to generate coherent and contextually relevant text. |
| Natural Language Prompts | Natural Language Prompts are human-readable instructions or requests provided to guide generative AI models, such as Large Language Models. Instead of using specific programming languages or commands, users can interact with these models by entering prompts in a more conversational or natural language form. The models then generate output based on the provided prompt. |
| 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. |

5

# Use Cases

## Configuration Example: Network Isolation by Subnet

**Use Case**

Acme Company has decided to use the dedicated infrastructure feature of Autonomous Database on Dedicated Exadata Infrastructure for its internal project teams and lines of business. The Acme I.T. department will take on the role of fleet administrator, creating and managing all Exadata Infrastructure, Autonomous Exadata VM Cluster, and Autonomous Container Database resources for the company. Each project team or line of business will have users who take on the database administrator role for their organization, creating autonomous databases for their database users (application developers, testers and deployers).

Acme I.T. will allocate resources to the organizations, making sure to provide autonomous container databases that meet the SLAs required by them. Additionally, in order to control the allocation of the resources fairly, Acme I.T. does not want any project team or line of business to have management access to the underlying dedicated infrastructure. Even further, Acme management gets audited by regulators, so it does not want Acme I.T. to be able to access the data that belongs to the different project teams or lines of business; that is, the data they are putting into their application databases.

Two project teams are the first to come on board: Roadrunner and Coyote. Neither the Roadrunner project team nor the Coyote project team wants the other team to be able to access its work in any way.

**Resulting Resources Needed**

- 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 will have an Internet Gateway resource that blocks all incoming internet traffic. This VCN will be named **DatabaseVCN**.
- Three private subnets in the VCN to provide network access isolation, one for Autonomous Database resources, and one for each team to use for their client and mid-tier resources. These subnets will be named **ITSubnet**, **RoadrunnerSubnet** and **CoyoteSubnet**.
- Three compartments to provide resource isolation, one for the resources Acme I.T. creates and the private subnet those resources will use, and one each for the Roadrunner teams and Coyote teams for its autonomous databases and private subnet. These compartments will be named **AcmeContainerDBs**, **RoadrunnerDBs** and **CoyoteDBs**.
- Three groups to which users can be assigned, one each for Acme I.T., the Roadrunner team and the Coyote team. These groups will be named **AcmeFAs**, **RoadrunnerDBAs** and **CoyoteDBAs**.

- Three policies to specify user access to the resources in each of the compartments. These policies will be named **AcmeContainerDBPolicy**, **RoadrunnerDBPolicy** and **CoyoteDBPolicy**.
- Two autonomous container databases, one that will be patched using *Next RU (next release update)* to provide stability and one that will be patched using *Latest RU (latest release update)* to provide access to the latest Autonomous Database features. These autonomous container databases will be named **StableContainerDB** and **LatestContainerDB**.

### High-Level Steps

1. The security administrator for Acme Company's cloud tenancy creates the **AcmeContainerDBs**, **RoadrunnerDBs** and **CoyoteDBs** compartments.
2. The network administrator for Acme Company's cloud tenancy creates the **DatabaseVCN** VCN and the **ITSubnet** subnet in the AcmeContainerDBs compartment, and then creates the **RoadrunnerSubnet** and **CoyoteSubnet** subnets in the **RoadrunnerDBs** and **CoyoteDBs** compartments, respectively.
3. The security administrator creates the **AcmeFAs**, **RoadrunnerDBAs**, and **CoyoteDBAs** groups.
4. The security administrator creates the **AcmeContainerDBPolicy**, **RoadrunnerDBPolicy**, and **CoyoteDBPolicy** policies.
5. After the network administrator finishes creating DatabaseVCN and ITSubnet, the security administrator adds the cloud user of a designated Acme I.T. member to the AcmeFAs group.
6. The newly authorized fleet administrator creates an Exadata Infrastructure resource in the AcmeContainerDBs compartment.
7. The fleet administrator then create an Autonomous Exadata VM Cluster resource in the AcmeContainerDBs compartment, specifying DatabaseVCN and ITSubnet as its VCN and subnet.
8. The fleet administrator creates the **StableContainerDB** and **LatestContainerDB** Autonomous Container Database resources in the AcmeContainerDBs compartment, specifying the newly created Exadata Infrastructure and Autonomous Exadata VM Cluster as their underlying resources.
9. The security administrator adds designated cloud users to the RoadrunnerDBAs group, thus authorizing them as database administrators for the Roadrunner project team, and then repeats the process for CoyoteDBAs.

## Step 1. Create Compartments

In this step, the security administrator for Acme Company's cloud tenancy creates the AcmeContainerDBs, RoadrunnerDBs and CoyoteDBs 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 the VCN and Subnets

In this step, network administrator for Acme Company's cloud tenancy creates the DatabaseVCN VCN and the ITSubnet subnet in the AcmeContainerDBs compartment, and then creates the RoadrunnerSubnet and CoyoteSubnet subnets in the RoadrunnerDBs and CoyoteDBs compartments, respectively.

To perform this step, the network administrator 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, the network administrator 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.

When adapting these instructions, the network administrator 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:

- ITSecList: the basic security list for ITSubnet. It is used when the ITSubnet subnet is created.
- RoadrunnerSeclist: the basic security list for RoadrunnerSubnet. It is used when the RoadrunnerSubnet subnet is created.
- RoadrunnerToITSeclist: a security list to permit resources in RoadrunnerSubnet to access Roadrunner team autonomous databases in ITSubnet. Once created, this security list is added to the ITSubnet subnet.
- CoyoteSeclist: the basic security list for CoyoteSubnet. It is used when the CoyoteSubnet subnet is created.
- CoyoteToITSeclist: a security list to permit resources in CoyoteSubnet to access Coyote team autonomous databases in the ITSubnet. Once created, this security list is added to the ITSubnet subnet.

### Security Rules in the ITSecList Security List

Here are the ingress rules created in the ITSecList security list.

| Stateless | Source | IP Protocol | Source Port Range | Destination Port Range | Type and Code | Allows |
|---|---|---|---|---|---|---|
| No | 10.0.0.0/24 | ICMP | | | All | ICMP traffic for: All |
| No | 10.0.0.0/24 | UDP | All | All | | UDP traffic for ports: All |
| No | 10.0.0.0/24 | TCP | All | All | | TCP traffic for ports: All |

Here is the egress rule created in the ITSecList security list.

| Stateless | Destination | IP Protocol | Source Port Range | Destination Port Range | Type and Code | Allows |
|---|---|---|---|---|---|---|
| No | 0.0.0.0/0 | All Protocols | | | All | All traffic for all ports |

### Security Rules in the RoadrunnerSeclist Security List

Here are the ingress rules created in the RoadrunnerSeclist security list.

| Stateless | Source | IP Protocol | Source Port Range | Destination Port Range | Type and Code | Allows |
|---|---|---|---|---|---|---|
| No | 0.0.0.0/0 | TCP | All | 22 | | TCP traffic for ports: 22 SSH Remote Login Protocol |
| No | 0.0.0.0/0 | TCP | All | 80 | | TCP traffic for ports: 80 |
| No | 0.0.0.0/0 | TCP | All | 443 | | TCP traffic for ports: 443 HTTPS |
| No | 0.0.0.0/0 | TCP | All | 943 | | TCP traffic for ports: 943 |
| No | 0.0.0.0/0 | UDP | All | 1194 | | UDP traffic for ports: 1194 |

Here is the egress rule created in the RoadrunnerSeclist security list.

| Stateless | Destination | IP Protocol | Source Port Range | Destination Port Range | Type and Code | Allows |
|---|---|---|---|---|---|---|
| No | 0.0.0.0/0 | All Protocols | | | All | All traffic for all ports |

### Security Rules in the RoadrunnerToITSeclist Security List

Here are the ingress rules created in the RoadrunnerToITSeclist security list. Note that the source CIDR is 10.0.10.0/24, which is the CIDR of RoadrunnerSubnet.

| Stateless | Source | IP Protocol | Source Port Range | Destination Port Range | Type and Code | Allows |
|---|---|---|---|---|---|---|
| No | 10.0.10.0/24 | TCP | All | 1521 | | TCP traffic for ports: 1521 |
| No | 10.0.10.0/24 | TCP | All | 2484 | | TCP traffic for ports: 2484 |
| No | 10.0.10.0/24 | TCP | All | 443 | | TCP traffic for ports: 443 HTTPS |
| No | 10.0.10.0/24 | TCP | All | 6200 | | TCP traffic for ports: 6200 |

No egress rules are created in the RoadrunnerToITSeclist security list.

### Security Rules in the CoyoteSeclist Security List

Note that the CoyoteSeclist security list has the same set of security rules as does RoadrunnerSeclist. While a single security list could be created and used for both project teams, the network administrator has created separate security lists in case one of the project teams requires additional security rules.

Here are the ingress rules created in the CoyoteSeclist security list.

| Stateless | Source | IP Protocol | Source Port Range | Destination Port Range | Type and Code | Allows |
|---|---|---|---|---|---|---|
| No | 0.0.0.0/0 | TCP | All | 22 | | TCP traffic for ports: 22 SSH Remote Login Protocol |
| No | 0.0.0.0/0 | TCP | All | 80 | | TCP traffic for ports: 80 |
| No | 0.0.0.0/0 | TCP | All | 443 | | TCP traffic for ports: 443 HTTPS |
| No | 0.0.0.0/0 | TCP | All | 943 | | TCP traffic for ports: 943 |
| No | 0.0.0.0/0 | UDP | All | 1194 | | UDP traffic for ports: 1194 |

Here is the egress rule created in the CoyoteSeclist security list.

| Stateless | Destination | IP Protocol | Source Port Range | Destination Port Range | Type and Code | Allows |
|---|---|---|---|---|---|---|
| No | 0.0.0.0/0 | All Protocols | | | All | All traffic for all ports |

### Security Rules in the CoyoteToITSeclist Security List

Here are the ingress rules created in the CoyoteToITSeclist security list. Note that the source CIDR is 10.0.11.0/24, which is the CIDR of CoyoteSubnet.

| Stateless | Source | IP Protocol | Source Port Range | Destination Port Range | Type and Code | Allows |
|---|---|---|---|---|---|---|
| No | 10.0.11.0/24 | TCP | All | 1521 | | TCP traffic for ports: 1521 |
| No | 10.0.11.0/24 | TCP | All | 2484 | | TCP traffic for ports: 2484 |
| No | 10.0.11.0/24 | TCP | All | 443 | | TCP traffic for ports: 443 HTTPS |
| No | 10.0.11.0/24 | TCP | All | 6200 | | TCP traffic for ports: 6200 |

No egress rules are created in the CoyoteToITSeclist security list.

## Step 3. Create Groups

In this step, the security administrator creates the AcmeFAs, RoadrunnerDBAs and CoyoteDBAs 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 4. Create Policies

In this step, the security administrator creates the AcmeContainerDBPolicy, RoadrunnerDBPolicy and CoyoteDBPolicy 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 **AcmeContainerDBPolicy** policy, the security administrator:

1. Sets the Compartment in the side menu to AcmeContainerDBs ***before*** clicking **Create Policy**.
2. Adds these Policy Statements:
   - Allow group AcmeFAs to MANAGE autonomous-exadata-infrastructures in compartment AcmeContainerDBs
   - Allow group AcmeFAs to MANAGE autonomous-container-databases in compartment AcmeContainerDBs
   - Allow group AcmeFAs to USE virtual-network-family in compartment AcmeContainerDBs
   - Allow group AcmeFAs to USE tag-namespaces in tenancy
   - Allow group RoadrunnerDBAs to READ autonomous-container-databases in compartment AcmeContainerDBs
   - Allow group CoyoteDBAs to READ autonomous-container-databases in compartment AcmeContainerDBs

When following these instructions for the **RoadrunnerDBPolicy** policy, the security administrator:

1. Sets the Compartment in the side menu to RoadrunnerDBs ***before*** clicking **Create Policy**.
2. Adds these Policy Statements:
   - Allow group RoadrunnerDBAs to MANAGE autonomous-databases in compartment RoadrunnerDBs

- Allow group RoadrunnerDBAs to MANAGE autonomous-backups in compartment RoadrunnerDBs
- Allow group RoadrunnerDBAs to USE virtual-network-family in compartment RoadrunnerDBs
- Allow group RoadrunnerDBAs to MANAGE instance-family in compartment RoadrunnerDBs
- Allow group RoadrunnerDBAs to USE tag-namespaces in tenancy

When following these instructions for the **CoyoteDBPolicy** policy, the security administrator:

1. Sets the Compartment in the side menu to CoyoteDBs ***before*** clicking **Create Policy**.
2. Adds these Policy Statements:
   - Allow group CoyoteDBAs to MANAGE autonomous-databases in compartment CoyoteDBs
   - Allow group CoyoteDBAs to MANAGE autonomous-backups in compartment CoyoteDBs
   - Allow group CoyoteDBAs to USE virtual-network-family in compartment CoyoteDBs
   - Allow group CoyoteDBAs to MANAGE instance-family in compartment CoyoteDBs
   - Allow group CoyoteDBAs to USE tag-namespaces in tenancy

## Step 5. Assign Fleet Administrators

In this step, the security administrator adds the cloud user of a designated Acme I.T. member to the AcmeFAs group.

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 in the AcmeContainerDBs compartment.

## Step 7. Create the Autonomous Exadata VM Cluster Resource

In this step, the fleet administrator follows the instructions in Create an Autonomous Exadata VM Cluster to create an Autonomous Exadata VM Cluster Infrastructure resource in the AcmeContainerDBs compartment, specifying DatabaseVCN and ITSubnet as its VCN and subnet.

## Step 8. Create Autonomous Container Database Resources

In this step, the fleet administrator follows the instructions in Create an Autonomous Container Database to create the StableContainerDB and LatestContainerDB Autonomous Container Database resources in the AcmeContainerDBs compartment, specifying the newly created Exadata Infrastructure and Autonomous Exadata VM Cluster as their underlying resources. When creating StableContainerDB, the fleet administrator configures its database maintenance version to *Next RU (next release update).* When creating LatestContainerDB

the fleet administrator configures its database maintenance version to *Latest RU (latest release update)* and changes the Backup retention policy from its default of 7 Days to 30 Days.

### Step 9. Assign Database Administrators

In this step, the security administrator adds designated cloud users to the RoadrunnerDBAs group, thus authorizing them as database administrators for the Roadrunner project team, and then repeats the process for CoyoteDBAs.

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.

## 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 Adminstrators 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, hover over **Application Integration** to expand its submenu, and then select **Notifications**.

   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 Subsciption 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 **Application Integration** to expand its submenu, and then select **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.

**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 Database Dedicated for Fleet Administrators Workshop:

- Lab 2: Provisioning a Cloud Exadata Infrastructure for Autonomous Database in OCI
- Lab 4: Provisioning Exadata Infrastructure for Autonomous database on Exadata Cloud@Customer

**Procedure**

1. Go to Autonomous Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Autonomous Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous Database resource types, click **Exadata Infrastructure**.

   The list of Exadata Infrastructure resources in your current Compartment is displayed.
3. In the side menu's **Compartment** list, 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.
- **Exadata system model:** Choose the type of Oracle Exadata Database Machine you want the 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 and X10M systems on 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 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, configure the automatic maintenance schedule by clicking **Modify 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. For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable.

  When you are finished, click **Update Maintenance Schedule**.
- **Contact Email(s):** You can provide up to ten contacts' email address for operational notifications and announcements. Click **Add Contact** after entering each email address.

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 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. Click **Create 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.

# 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 Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Autonomous Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous Database resource types, click **Exadata Infrastructure**.

   The list of Exadata Infrastructure resources in your current Compartment is displayed.
3. In the side menu's **Compartment** list, 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 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 |
|---|---|
| View or copy the resource's OCID | On the **Exadata Infrastructure Information** tab, click **Show or Copy** in the **OCID** field. |
| Edit the resource's maintenance schedule | On the **Exadata Infrastructure Information** tab, click **Edit** in the **Maintenance Schedule** field. |
| View upcoming and past maintenance | On the **Exadata Infrastructure Information** tab, click **View** in the **Next Maintenance** field. A list of upcoming planned maintenance activities, if any, is displayed. To view past maintenance activities, click **Maintenance History** in the side menu. |

| Action | Steps |
|---|---|
| Scale up the resource | Click **Scale Infrastructure** and enter new values for Database Servers and/or Storage Servers choosing the appropriate option. Then, click **Scale**.<br>You can only enter a value that is greater than the current value of the database or storage servers for the Exadata Infrastructure resource.<br><br>**Note:**<br>You can not scale an Exadata Infrastructure resource with an Autonomous Exadata VM Cluster created in it. |
| Add tags to the resource | Click **Add Tags**. |
| Move the resource to a different compartment | Click **Move Resource** and choose the new compartment for the Exadata Infrastructure resource. |
| Terminate the resource | Click **Terminate** and follow the instructions given in Terminate an Exadata Infrastructure Resource. |
| Update the display name | Click **Update Display Name**. Enter the new and current display names, and click **Update Display Name**. |
| View the tags applied to the resource | Click the **Tags** tab. |

At the bottom of the page, a list of Autonomous Exadata VM Clusters in the selected Compartment that use the chosen Exadata Infrastructure resource is displayed. Using this list you can perform a variety of actions.

| Action | Steps |
|---|---|
| Create an Autonomous Exadata VM Cluster | Click **Create Autonomous 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 of the one you want to move 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 of the one you want to apply tags to and then click **Add Tags**. |

| Action | Steps |
| --- | --- |
| 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. |
| 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.

> **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. Click **Move Resources**.
3. In the **Move Resource to a Different Compartment** page, select the new compartment.
4. Click **Move Resource**.

# 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 whose maintenance schedule you want to change.

   For instructions, see View Details of an Exadata Infrastructure Resource.
2. Click **Terminate**.
3. In the **Terminate Exadata Infrastructure** dialog:
   - Read the warning about terminating an Exadata Infrastructure resource.
   - Enter the name of the Exadata Infrastructure resource.
   - Click **Terminate Exadata Infrastructure** to begin the termination process.

# View a List of Exadata Infrastructure Resources

You view a list of Exadata Infrastructure resources bu going to the **Exadata Infrastructure** page.

### Required IAM Policies

```
inspect cloud-exadata-infrastructures
```

### Procedure

1. Go to Autonomous Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Autonomous Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous Database resource types, click **Exadata Infrastructure**.

   The Exadata Infrastructure page is displayed, showing the list of Exadata Infrastructure resources in your current compartment. To see the list of a different compartment, choose its name from the side menu's compartment list.

### 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. |
| 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 **Add Tags**. |
| View the tags applied to a listed Exadata Infrastructure resource | In the list of Exadata Infrastructure resources, click the menu in the row of the resource whose tags you want to view and then click **View Tag**. |
| 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 and then click **Terminate**. Then, in the Terminate Exadata Infrastructure dialog:<br>• Read the warning about terminating an Exadata Infrastructure resource.<br>• Enter the name of the Exadata Infrastructure resource.<br>• Click **Terminate Exadata Infrastructure** to begin the termination process. |

# 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. Click the **Edit** link in the **Maintenance schedule** field.
3. On the **Edit Automatic Maintenance** dialog, 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 infrastructure maintenance.

  > **Tip:**
  >
  > For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable.

4. When finished, click **Update Maintenance Schedule**.

# 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 the **View** link in the **Next Maintenance** field.

   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.

# 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 the **View** link in the **Next Maintenance** field.

   The **Exadata Infrastructure Maintenance** page lists any upcoming planned maintenance.
3. Click **Maintenance History** in the side menu 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:**
>
> 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 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.

## Prerequisites

**IAM Policy Requirements**

| Deployment Choice | IAM Policies |
|---|---|
| Oracle Public Cloud | `manage cloud-autonomous-vmclusters`<br>`use cloud-exadata-infrastructures` |
| Exadata Cloud@Customer | `manage autonomous-vmclusters`<br>`use exadata-infrastructures` |

**Resource Requirements**

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

- 10 OCPUs or 40 ECPUs
- 200GB memory
- 700GB local storage
- 3.25TB Exadata storage

### Network Requirements

#### Subnet Sizes

- **Oracle Public Cloud Deployments:**
  If you are provisioning an Autonomous Exadata VM Cluster resource in more than one region, make sure the IP address space of the Virtual Cloud Networks (VCNs) does not overlap. This is important if you want to set up disaster recovery with Autonomous Data Guard.

  The two subnets you create for the Autonomous Database service on Oracle Cloud must not overlap with 192.168.128.0/20. The following table lists the minimum required subnet sizes for Autonomous 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.

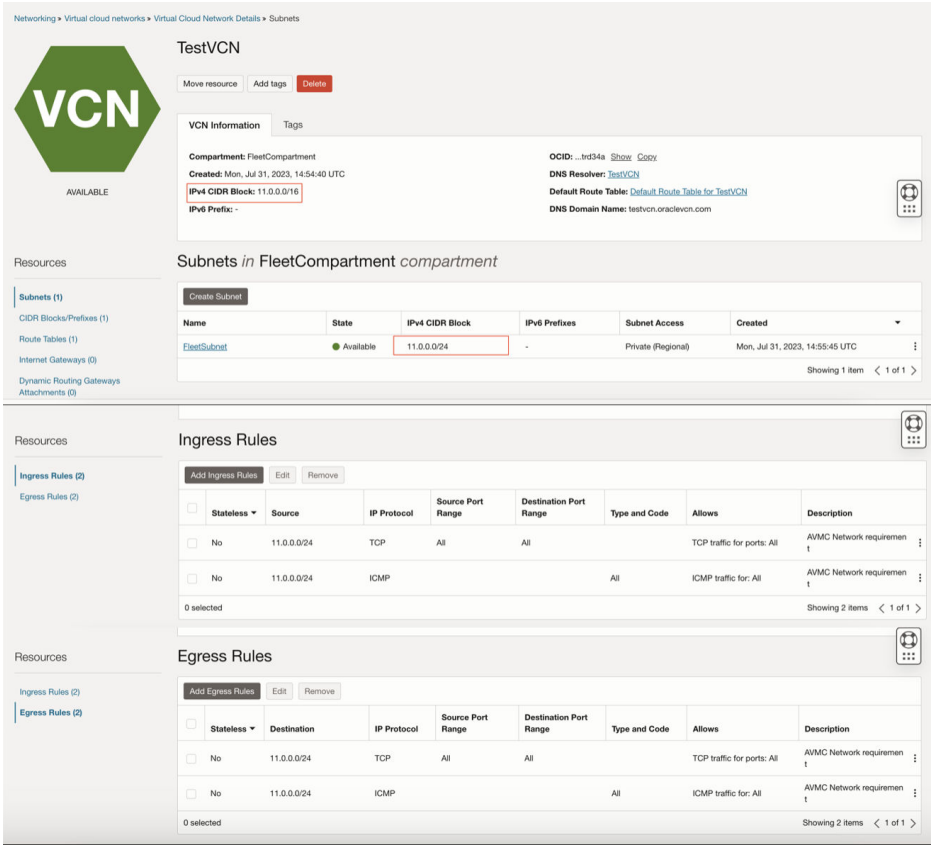
- **Exadata Cloud@Customer Deployments:**
  For Exadata Cloud@Customer IP requirements, refer to IP Addresses and Subnets for Exadata Cloud at Customer.

#### Security Lists

- **Oracle Public Cloud 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.




– Optionally, depending on your requirement, you may also need to open the following ports to connect and operate Autonomous 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<br>Fast Application Notification (FAN) traffic |
| 443 | TCP | Oracle Application Express<br>SQL Developer Web |

- **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 database and standby database in an Autonomous Data Guard setup.

## Procedure

> **Tip:**
>
> For "try it out" alternatives that demonstrate these instructions, see the following labs in Oracle Autonomous Database Dedicated for Fleet Administrators Workshop:
>
> - Lab 3: Provisioning a Cloud Autonomous Exadata VM Cluster for Autonomous Database on Dedicated Infrastructure
> - Lab 5: Provisioning an Autonomous VM Cluster on Exadata Cloud@Customer

1. Go to Autonomous Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Autonomous Database in the Oracle Cloud Infrastructure Console.
2. In the side menu's list of Autonomous Database resource types, click **Autonomous Exadata VM Cluster**.

   The list of Autonomous Exadata VM Clusters in your current Compartment is displayed.
3. In the side menu's **Compartment** list, 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.<br>**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. | |

| Setting | Description | Notes |
|---|---|---|
| **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.<br>Click **Change compute model** if you wish to select OCPU. OCPU compute model is based on the physical core of a processor with hyper-threading enabled. | The compute model chosen here will apply to all the Autonomous Container Databases and Autonomous Databases created in this Autonomous Exadata VM Cluster resource. Refer to Compute Models. |
| **Autonomous VM Cluster resources: DB Server Selection** | Lists the DB Servers (VMs) used to deploy the new Autonomous Exadata VM Cluster (AVMC) resource.<br>The maximum resources (CPUs, Memory, and Local Storage) available per VM is also displayed.<br>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:<br>• Available CPUs<br>• Available memory (GB)<br>• Available local storage (GB)<br>• Number of VM Clusters and Autonomous VM Clusters in it.<br>By default, all the DB Servers that have the minimum resources needed to create an Autonomous VM Cluster are selected. You can remove a 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.<br>After selecting the DB Servers, click **Save Changes**. | 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.<br>**Note:**<br>You can not add or remove DB Servers after provisioning an AVMC. |

| 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.<br>**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 2 available OCPUs or 8 available ECPUs per node.<br>**CPU count per VM or node:** Specify the CPU count for each individual VM. The minimum value is 5 OCPUs or 20 ECPUs per VM.<br>**Database memory per CPU (GB):** The memory per CPU allocated for the Autonomous Databases in the Autonomous VM Cluster.<br>**Allocate Storage for Local Backups:** For Exadata Cloud@Customer, you can check this option to configure the Exadata storage to enable local database backups.<br>**Database storage (TB):** Data storage allocated for Autonomous Database creation in the Autonomous VM Cluster. | VM and node are used interchangeably between Oracle Exadata Cloud@Customer and Oracle Public Cloud deployments.<br>As you configure the resource parameters, the aggregate resources needed to create the Autonomous Exadata VM Cluster and the formulae used to calculate these values are displayed on the right side of the resource configuration section.<br>The total resources requested for Autonomous Exadata VM Cluster are calculated using the below formulae:<br>**CPU Count** = CPU Count per Node x Node Count<br>**Memory** = [CPU Count per Node x Database Memory per CPU + Internal Database Service Memory: (40GB)] x Node Count<br>**Local Storage** = Internal Local Service Storage: ([Autonomous VM Cluster storage: xxGB] + [Autonomous Container Database storage: 100GB + 50GB x ACD Count] x 1.03) x Node Count<br>In the above formula, *Autonomous VM Cluster storage* value depends on the *Exadata system model* used, as follows:<br>• X7-2 and X8-2 systems: 137GB<br>• X8M-2 & X9M-2 systems: 184GB<br>**Exadata Storage** = [User Data Storage + Internal Database Service Storage: ( 200GB + 50GB x ACD Count x Node Count )] x 1.25 |

| 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.<br>**Subnet:** A subnet within the above selected VCN for the new Autonomous Exadata VM Cluster.<br>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.<br>Click **+ Another Network Security Group** to add more network security groups. | APPLIES TO: Oracle Public Cloud only |
| **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, configure the automatic maintenance schedule by clicking **Modify Schedule**.<br>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.<br>When you are finished, click **Update Maintenance Schedule**. | For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable. |
| **License Type** | The license type you wish to use for the new Autonomous Exadata VM Cluster.<br>You have the following options:<br>**Bring your own license:** If you choose this option, make sure you have proper entitlements to use for new service instances that you create.<br>**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. |

| Setting | Description | Notes |
|---|---|---|
| **Show/hide advanced options** | By default, advanced options are hidden. Click **Show advanced options** to display them. | |
| **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 `Java.util.TimeZone` class and the Oracle Linux operating system.<br>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**).<br>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.<br>The port numbers 1522, 1525, 5000, 6100, 6200, 7060, 7070, 7879, 8080, 8181, 8888, and 8895 are **reserved** for special purposes and *can not be used* as Non-TLS or TLS port numbers.<br>The SCAN listener ports **can not be changed** after provisioning the AVMC resource.<br>As ORDS certificates are one-way TLS certificates, choosing between one-way TLS and mutual TLS (mTLS) is applicable only to **database TLS certificates**. |
| **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 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. Click **Create Autonomous Exadata VM Cluster**.

# 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 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 Database on Exadata Cloud@Customer:**

```
inspect autonomous-vmclusters
```

**For Autonomous 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 Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Autonomous Database in the Oracle Cloud Infrastructure Console.

2. In the side menu's list of Autonomous Database resource types, click **Autonomous Exadata VM Cluster**.

   The Autonomous Exadata VM Clusters page is displayed, showing the list of Autonomous Exadata VM Clusters in your current Compartment. To see the list for a different Compartment, choose its name from the side menu's Compartment list.

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. Scroll to the bottom of the **Details** page of the Exadata Infrastructure resource.

   Click **Autonomous Exadata VM Clusters** under **Resources** to list the Autonomous Exadata VM Clusters 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 | Click **Move Resource** and choose the new compartment for the Autonomous Exadata VM Cluster resource. |
| 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 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 **Add 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:<br>• Read the warning about terminating an Autonomous Exadata VM Cluster.<br>• Enter the name of the Autonomous Exadata VM Cluster.<br>• Click **Terminate Autonomous Exadata VM Cluster** to begin 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 Database on Exadata Cloud@Customer:**

```
inspect autonomous-vmclusters
```

**For Autonomous Database on Oracle Public Cloud:**

```
inspect cloud-autonomous-vmclusters
```

### Procedure

1. Go to Autonomous Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Autonomous Database in the Oracle Cloud Infrastructure Console.

2. In the side menu's list of Autonomous Database resource types, click **Autonomous Exadata VM Cluster**.

   The list of Autonomous Exadata VM Clusters in your current Compartment is displayed.

3. In the side menu's **Compartment** list, 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 the name of the Autonomous Exadata VM Cluster whose details you want to view.

   The **Details** page for the chosen Autonomous Exadata VM Cluster is displayed.

### 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 | On the **Autonomous Exadata VM Cluster information** tab, click the name of the Exadata Infrastructure resource in the Exadata Infrastructure field. |
| View or copy the Autonomous Exadata VM Cluster's OCID | On the **Autonomous Exadata VM Cluster information** tab, click **Show or copy** in the OCID field. |
| Edit the Autonomous Exadata VM Cluster's maintenance schedule | On the **Autonomous Exadata VM Cluster information** tab, click **Edit** in the **Maintenance schedule** field. |
| View upcoming and past maintenance | On the **Autonomous Exadata VM Cluster information** tab, click **View** in the **Next maintenance** field. A list of upcoming planned maintenance activities, if any, is displayed for both the Autonomous Exadata VM Cluster and the Exadata Infrastructure resource containing it. To view past maintenance activities, click **Maintenance history** in the side menu. |
| Terminate the Autonomous Exadata VM Cluster | Click **Terminate** and follow the instructions in Terminate an Autonomous Exadata VM Cluster. |
| Add tags to the Autonomous Exadata VM Cluster | Click **Add tags**. |
| View the tags applied to the Autonomous Exadata VM Cluster | Click the **Tags** tab. |
| Manage Certificates | Click **Manage certificates** and follow instructions outlined in Manage Security Certificates for an Autonomous Exadata VM Cluster Resource. |

| Action | Steps |
|---|---|
| Move Autonomous Exadata VM Cluster to a different compartment | Click **Move Resource**. On the **Move Resource to a Different Compartment** dialog, choose the new compartment and click **Move Resource**. |
| View resource allocation details | Click **View Details** under **Resource allocation**. For additional information, see Resource Usage Tracking. |
| Scale up or scale down the resources within the Autonomous Exadata VM Cluster | Click **Scale Autonomous VM Cluster** and follow the instructions in Manage Autonomous Exadata VM Cluster Resources. |

### Resources Listed

As you scroll down, the following resources associated with this AVMC are listed on 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 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 under Resources. 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 **Add tags**. |
| Move a listed Autonomous Container Database to a different compartment | Click **Move resource**. On the **Move resource to a different compartment** dialog, choose the new compartment and click **Move Resource**. |

| Actions | Steps |
|---|---|
| Restart a listed Autonomous container Database resulting in a rolling restart of all Autonomous Databases in this container. | 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 Exadata VM Cluster** dialog:<br>• Read the warning about terminating an Autonomous Exadata VM Cluster.<br>• Enter the name of the Autonomous Exadata VM Cluster.<br>• Click **Terminate Autonomous Exadata VM Cluster** to begin the termination process. |

## Change the License Type on an Autonomous VM Cluster

APPLIES TO: Exadata Cloud@Customer only

Follow these steps to update the license type of an autonomous Exadata VM cluster:

### Required IAM Policies

```
use autonomous-vmclusters

inspect exadata-infrastructures
```

### Procedure

1. Open the navigation menu. Under **Database**, click **Exadata 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. 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 Changes**.

## Move an Autonomous Exadata VM Cluster to Another Compartment

Follow these steps to move an Autonomous Exadata VM cluster from one compartment to another.

> **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 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 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. Click **Move Resource**.
3. Select the new compartment.
4. Click **Move Resource**.

## 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 Database listener from the **Details** page of an Autonomous Exadata VM Cluster (AVMC).

**Required IAM Policies**

**For Autonomous Database on Exadata Cloud@Customer**:

```
manage autonomous-vmclusters
```

**For Autonomous 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:<br>**Database SSL certificate:** Choose this option to manage SSL certificate for Database client connections.<br>**ORDS SSL certificate:** Choose this option to manage ORDS SSL certificate for Application Express (APEX) applications. | Rotating a *Database SSL certificate* is a disruptive operation and may cause databases to become unavailable to new connections *for the duration of the certificate update*.<br>Refer to Certificate Management for requirements about downloading and replacing the client wallets.<br>Rotating an *ORDS SSL certificate* 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**.<br>Optionally, you can choose **Bring your own certificate** and provide the following details:<br>**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.<br>**Specify CA certificate:** Optionally, you can specify a CA certificate with the following options:<br>• **Certificate authority:** If this option is selected, select a certificate authority from the select list, changing compartments if necessary.<br>• **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.<br>These certificates must be signed and must be in the PEM format, that is, their file extension must be .pem, .cer, or .crt only. |

4. Click **Save changes**.
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 Database on Exadata Cloud@Customer

```
inspect autonomous-vmclusters
```

For Autonomous 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 Database you want to monitor. For instructions, see View Details of an Autonomous Database on Dedicated Exadata Infrastructure.
- On the **Autonomous Exadata VM cluster information** tab, locate the **Resource Allocation** section. In this 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 Database Storage (TB)
  - Total Autonomous 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 Database Storage in TB
      - Autonomous 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 Database.
  - The number of CPUs provisioned to Autonomous 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 Database on Exadata Cloud@Customer:**

```
manage autonomous-vmclusters
```

**For Autonomous 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. 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 Databases created in that AVMC.

4. Click **Save 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 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 Database on Exadata Cloud@Customer:**

```
manage autonomous-vmclusters
```

**For Autonomous 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 move. See View Details of an Autonomous Exadata VM Cluster.
2. Click **Terminate**.
3. Confirm that you wish to terminate your Autonomous Exadata VM Cluster in the confirmation dialog.
4. Click **Terminate VM Cluster**.

# Manage Exadata Infrastructure Maintenance

Oracle schedules and performs all patching and other maintenance operations on all the infrastructure resources of an Autonomous 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 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, click **Update Maintenance Schedule**.

## 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 Database on Exadata Cloud@Customer**

```
use autonomous-vmclusters

inspect exadata-infrastructures
```

**For Autonomous 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 the **View** link in the **Next Maintenance** field.

   The **Autonomous Exadata VM Cluster Maintenance** page lists any upcoming planned maintenance.

   For more information about the details displayed in the **Autonomous Exadata VM Cluster Maintenance** page, see Scheduled Maintenance Details.
3. For each maintenance event listed in the **Autonomous Exadata VM Cluster 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 Database on Exadata Cloud@Customer**

```
inspect autonomous-vmclusters
```

**For Autonomous 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 the **View** link in the **Next Maintenance** field.

   The **Autonomous Exadata VM Cluster Maintenance** page lists any upcoming planned maintenance.
3. Click **Maintenance History** in the side menu to see a list of past maintenance.

# Create and Manage Autonomous Container Databases

## 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 Database on Dedicated Exadata Infrastructure. ACDs are provisioned inside an Autonomous Exadata VM Cluster (AVMC) and serve as containers for one or more Autonomous 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 Databases. To gain a comprehensive understanding of the four-layer architecture used with the an Autonomous Database on Dedicated Exadata Infrastructure and to understand the positioning of ACD within this architecture, refer to Components of Autonomous Database on Dedicated Exadata Infrastructure.

ACDs provide the benefit of operating in isolation from each other, allowing you to separate Autonomous 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 Databases. See User Roles Associated with Autonomous 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:

- 2 OCPUs or 8 ECPUs
- 50GB local storage

> **Note:**
>
> 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.

## Database Features Managed from Autonomous Container Database

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

| Autonomous Database Feature | Notes | Further Reference |
|---|---|---|
| **Database Version**<br>You can set the database home version while provisioning an ACD. | You can choose either the latest Oracle Home Version or its immediate predecessor.<br>For example: Suppose the latest Oracle Database version supported by Autonomous Database is 19.22.0.1.0. Then, the **Select base image** drop-down lists 19.22.0.1.0 and 19.21.0.1.0. for you to choose. | - |

| Autonomous Database Feature | Notes | Further Reference |
|---|---|---|
| **Autonomous Data Guard**<br>Configuring Autonomous Data Guard enables you to keep your critical production databases available to mission critical applications despite failures. | You can configure Autonomous Data Guard to create primary and standby ACDs.<br>The primary and secondary ACDs can also be deployed in different regions (cross-region). However, in a cross-region Autonomous Data Guard setup, note the following about encryption keys:<br>• If you are managing keys using the OCI Vault service, you must use a key from a virtual private vault located in the region of the primary database. This vault must be replicated in the region of the standby database. See Replicating Vaults and Keys for information on creating and managing these resources. The replicated vault used by the standby is read-only. So, when the standby assumes the primary role from a switchover or failover, you cannot create a new Pluggable Database or rotate the key.<br>• Minimum fast start lag limit value supported by ACDs with 19.17 and older versions is 30 seconds.<br>• You can enable Autonomous Data Guard for both new and existing ACDs. | Protect Critical Databases from Failures and Disasters Using Autonomous Data Guard |

| Autonomous Database Feature | Notes | Further Reference |
|---|---|---|
| **Maintenance Schedule**<br>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.<br>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 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.<br>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.<br>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:<br>• Already undergoing a maintenance update, or<br>• Scheduled for a maintenance activity simultaneously as the ACD.<br>You may experience downtime for your ACD and the associated Autonomous Databases depending on the configuration of your ACD's maintenance schedule. | Service Maintenance |

| Autonomous Database Feature | Notes | Further Reference |
|---|---|---|
| **Backup Retention Policy**<br>To support high availability, Autonomous Database automatically backs up your database for you. The retention period for backups is up to 60 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.<br>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.<br>On Oracle Public Cloud deployments, the backup retention policy value defaults to 15 days.<br>On Exadata Cloud@CustomerExadata Cloud@Customer deployments, the backup retention policy:<br>• Defaults to 30 days for Object Storage and Network File System (NFS) backup destination types.<br>• Is controlled by the Recovery Appliance protection policy for the Recovery Appliance backup destination type.<br>All the backups are automatically deleted after the backup retention period. | Backup and Restore Autonomous Databases. |

| Autonomous Database Feature | Notes | Further Reference |
|---|---|---|
| **Backup Destination**<br>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. | APPLIES TO: Exadata Cloud@Customer only<br>As of the current release, the backup destination type can only be set while enabling automatic backups on an ACD and can not be changed later.<br>The possible options are:<br>• **Object Storage:** Stores backups in an Oracle-managed object storage container on Oracle Cloud Infrastructure.<br>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.<br>• **Network File System (NFS):** Stores backups in a Network File System (NFS) storage location.<br>If you choose Network File System (NFS) as the type, select a previously defined Backup Destination that uses Network File System (NFS) storage.<br>• **Recovery Appliance:** Stores backups in one of your previously defined backup destinations that use Oracle Zero Data Loss Recovery Appliance.<br>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 Autonomous Container Database, and the VPC user name password.<br>Provide the connection string that connects to the recovery appliance in an Oracle "easy connect" | To learn more about configuring NFS backup destination for Cloud@Customer, see Prerequisites for Backup Destinations for Exadata Cloud@Customer.<br>Refer to Edit Autonomous Container Database Backup Settings for instructions on changing the backup destination type after provisioning the ACD. |

| Autonomous Database Feature | Notes | Further Reference |
|---|---|---|
| | string format, that is, `<host>:<port>/<service name>`, where `<host>` is the SCAN hostname of the Zero Data Loss Recovery Appliance. | |
| **Resource Management Attributes**<br>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:<br>• **Database split threshold (CPU):** The CPU value beyond which an Autonomous Database will be opened across multiple nodes. The default value of this attribute is 16 for OCPUs and 64 for ECPUs.<br>• **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.<br>• **Distribution affinity:** Determines whether an Autonomous 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 Compute Management and Billing. |
| **Shared Server Connections**<br>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 Database Feature | Notes | Further Reference |
|---|---|---|
| **Encryption Key**<br>By default, Autonomous Database creates and manages all the master encryption keys used to protect your data, storing them in a secure PKCS 12 keystore on the same Exadata systems where the databases reside.<br>If your company security policies require, Autonomous 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.<br>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<br>Rotate the Encryption Key of 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 |
| 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 |
| Change the backup retention policy of an Autonomous Container Database | Edit Autonomous Container Database Backup Settings |
| Edit the maintenance preferences of an Autonomous Container Database | Update Autonomous Container Database Maintenance Preferences |
| Restart an Autonomous Container Database | Restart an Autonomous Container Database |
| 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 |
| Terminate an Autonomous Container Database | Terminate an Autonomous Container Database |

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

# 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 | `manage autonomous-container-databases`<br>`use cloud-exadata-infrastructures`<br>`use cloud-autonomous-vmclusters` |
| Exadata Cloud@Customer | `manage autonomous-container-databases`<br>`use cloud-exadata-infrastructures`<br>`use autonomous-vmclusters`<br>`use backup-destinations` |

### Minimum Resource Requirements

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

- 2 OCPUs or 8 ECPUs
- 50GB local storage

### Related Live Labs

For a "try it out" alternative that demonstrates these instructions, see *Lab 6: Provisioning an Autonomous Container Database* in Oracle Autonomous Database Dedicated for Fleet Administrators Workshop.

### Procedure

1. Go to Autonomous Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Dedicated Autonomous 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. In the side menu's **Compartment** list, select 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.<br>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.<br>**Tip:**<br>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. |
| **Exadata Infrastructure** | The Exadata Infrastructure to host the new Autonomous Container Database. | APPLIES TO: Oracle Public Cloud only |
| **Autonomous Exadata VM Cluster** | The Autonomous Exadata VM Cluster to host the new Autonomous Container Database. | |
| **Container Database Software Version** | Oracle Home Version for the Autonomous Container Database.<br>Select the **base image** version from the drop-down list. The latest Oracle Home Version is selected by default. | You can choose either the latest Oracle Home Version or its immediate predecessor.<br>For example: Suppose the latest Oracle Database version supported by Autonomous Database is 19.18.0.1.0. Then, the **Select base image** drop-down lists 19.18.0.1.0 and 19.17.0.1.0. for you to choose. |

| Setting | Description | Notes |
|---|---|---|
| **Autonomous Data Guard** | Select **Enable Autonomous Data Guard** to configure Autonomous Data Guard to create primary and standby autonomous container databases. | Configuring Autonomous Data Guard enables you to keep your critical production databases available to mission critical applications despite failures. |
| **Configure Autonomous Data Guard: Standby Container Database** | Provide information about the standby autonomous container database: its **compartment**, **display name**, **region**, the underlying **Exadata Infrastructure** resource, and **Autonomous Exadata VM Cluster** in which to place it. | If you choose to create an Autonomous Container Database with a **cross-region Autonomous Data Guard association**, note the following about encryption keys:<br>• If you are managing keys using the OCI Vault service, you must use a key from a virtual private vault located in the region of the primary database. This vault must be replicated in the region of the standby database. See Replicating Vaults and Keys for information on creating and managing these resources.<br>• The replicated vault used by the standby is read-only. So, when the standby assumes the primary role from a switchover or failover, you cannot create a new Pluggable Database or rotate the key. |
| **Configure Autonomous Data Guard: Protection mode** | Choose a protection mode from **Maximum Performance** and **Maximum Availability**. | **Maximum Performance** is selected by default.<br>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 Choosing Autonomous Data Guard Configuration Options. |

| Setting | Description | Notes |
| --- | --- | --- |
| **Configure Autonomous Data Guard: Automatic failover** | Optionally, select **Enable automatic failover** and choose a value for **Fast start failover lag limit** between *5 and 3600* seconds if the protection mode is Maximum Performance. | **Note:** Minimum fast start lag limit value supported by ACDs with 19.17 and older versions is 30 seconds.<br><br>For more details about automatic failover, see Automatic Failover or Fast-Start Failover. |
| **Configure automatic maintenance** | The **Configure automatic maintenance** panel displays the following default settings:<br>• **Maintenance method:** Rolling with time-zone file update disabled.<br>• **Container database maintenance version:** Next release update (RU)<br>• **Maintenance schedule:** No schedule preference specified.<br>Optionally, you can configure a maintenance preference or schedule by clicking **Modify Maintenance** 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.<br>Optionally, you can also select **Enable time-zone update**.<br>See Service Maintenance Types for more information.<br>**Configure container database maintenance version**: Choose either **Next RU** or **Latest RU** to configure the next maintenance version of the ACD.<br>**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**. For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable.<br>Click **Save** to close this dialog. | Updating the time-zone file would require a complete downtime for the ACD and the associated Autonomous Databases. The downtime is dependent on the amount of data that is time-zone sensitive.<br>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.<br>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.<br>While provisioning an ACD with Autonomous Data Guard, you can not disable the automatic 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.<br>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 destination type** | After enabling automatic backups, select a **Backup destination type** and then specify options based on the selected type.<br>The possible options are:<br>• **Object Storage:** Stores backups in an Oracle-managed object storage container on Oracle Cloud Infrastructure.<br>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.<br>• **Network File System (NFS):** Stores backups in a Network File System (NFS) storage location.<br>If you choose Network File System (NFS) as the type, select a previously defined Backup Destination that uses Network File System (NFS) storage.<br>• **Recovery Appliance:** Stores backups in one of your previously defined backup destinations that use Oracle Zero Data Loss Recovery Appliance.<br>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 Autonomous Container Database, and the VPC user name password.<br>Provide the connection string that connects to the recovery appliance in an Oracle "easy connect" string format, | APPLIES TO: Exadata Cloud@Customer only<br>As of the current release, the backup destination type can only be set while enabling automatic backups on an ACD and can not be changed later.<br>To learn more about configuring NFS backup destination for Cloud@Customer, see Prerequisites for Backup Destinations for Exadata Cloud at Customer.<br>Refer to Edit Autonomous Container Database Backup Settings to change the backup destination type after provisioning the ACD. |

| Setting | Description | Notes |
|---|---|---|
| | that is, `<host>:<port>/<service name>`, where `<host>` is the SCAN hostname of the Zero Data Loss Recovery Appliance. | |
| **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. | On Oracle Public Cloud deployments, the backup retention policy value defaults to 15 days.<br>On Exadata Cloud@Customer deployments, for Object Storage and Network File System (NFS) backup destination types, the backup retention policy value defaults to 30 days.<br>For the Recovery Appliance backup destination type, this value is controlled by the Recovery Appliance protection policy.<br>All the backups are automatically deleted after the backup retention period. |
| **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:<br>• **Database split threshold (CPU):** The CPU value beyond which an Autonomous Database will be opened across multiple nodes. The default value of this attribute is 16 for OCPUs and 64 for ECPUs.<br>• **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.<br>• **Distribution affinity:** Determines whether an Autonomous Database must be opened across a minimum or maximum of nodes. By default, Minimum nodes is selected with Maximum nodes being the other option.<br>Optionally, select the **Enable shared server connections** checkbox to support the Net services architecture. | If the Node failover reservation is set to 0%, your Autonomous Database may face a complete outage during VM failure and maintenance operations such as database patching and container database restart.<br>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.<br>Select **Encrypt using customer-managed keys** and then select either the **Vault** and **Master encryption key** or the **Key Store** to use for the Autonomous Container Database, depending on whether you are creating the container database on Oracle Public Cloud or on Exadata Cloud@Customer. | By default, **Encrypt using Oracle-managed keys** is selected.<br>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.<br>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. |

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. Click **Create 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.

# 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 Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Dedicated Autonomous Database in the Oracle Cloud Infrastructure Console.

2. In the side menu's list of Autonomous Database resource types, click **Autonomous Container Database**.

   The list of Autonomous Container Databases in your current Compartment is displayed.

3. In the side menu's **Compartment** list, 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.

4. Click the name of the Autonomous Container Database whose details you want to view.

The **Details** page for the chosen 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
  - 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
- 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 ACD's encryption key by clicking **Rotate encryption key** button and providing your confirmation on the confirmation dialog that appears.
- Edit the backup retention settings:
  - Exadata Cloud@Customer deployments: Click the **Edit backup settings** to launch the **Edit backup settings** dialog, change the settings, and click **Save changes**.
  - Oracle Public Cloud deployments: Change the backup retention policy (number of days) by clicking the **Edit** link next to **Backup retention policy** in the **Autonomous Container Database information** tab.
- Add tags by clicking the **Add tags** button under **More 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.
- Terminate an Autonomous Container Database

At the bottom of the page is a list of Autonomous Databases in the selected Compartment that use the chosen Autonomous Container Database is displayed. Using this list you can perform a variety of actions.

| Action | Steps |
|---|---|
| Create an Autonomous Database | Click **Create Autonomous Database** and follow the instructions listed in Create a Dedicated Autonomous Database. |
| View details of a listed Autonomous Database | In the list of Autonomous Databases, click the name of the one whose details you want to view. |
| Move the listed Autonomous Database to a different compartment | In the list of Autonomous Databases, click the menu in the row of the one you want to move to a different compartment and then click **Move resource**. |
| Copy the OCID of a listed Autonomous Database | In the list of Autonomous 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 Database | In the list of Autonomous Database, click the menu in the row of the one you want to apply tags to and then click **Add Tag(s)**. |
| Enable Autonomous Data Guard | Click **Enable Autonomous Data Guard** and follow the instructions in Enable Autonomous Data Guard on an Autonomous Container Database. |
| Terminate a listed Autonomous Database | In the list of Autonomous Databases, click the menu in the row of the one you want to terminate and then click **Terminate**. Then, in the **Terminate Database** dialog:<br>1. Read the warning about terminating an Autonomous Database.<br>2. Enter the name of the Autonomous Database.<br>3. Click **Terminate Database** to begin the termination process. |

# 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 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.

## 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 Databases created in it unavailable.

> **Note:**
>
> You may experience performance degradation while restarting an Autonomous Container Database with Database In-Memory enabled Autonomous 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.

## 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 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. Click **Move Resource**.
3. In the **Move Resource to a Different Compartment** page, select the new compartment.
4. Click **Move Resource**.

## 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 Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Dedicated Autonomous Database in the Oracle Cloud Infrastructure Console.

2. In the side menu's list of Autonomous 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. To see the list for a different Compartment, choose its name from the side menu's **Compartment** list.

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 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 and Billing.
   - CPU value of the largest provisionable Autonomous Database.
   - **Provisioned CPUs** or the total CPUs allocated for all the Autonomous 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. |

| Action | Steps |
|---|---|
| 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 **Add Tag(s)**. |
| 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 *Autonomous Container Database name* 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:<br>1. Read the warning about terminating an Autonomous Container Database.<br>2. Enter the name of the Autonomous Container Database.<br>3. Click **Terminate Autonomous Container** 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 Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Dedicated Autonomous Database in the Oracle Cloud Infrastructure Console.

2. In the side menu's list of Autonomous Database resource types, click **Autonomous Exadata VM Cluster**.

   The list of Autonomous Exadata VM Cluster resources in your current Compartment is displayed.

3. In the side menu's **Compartment** list, 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.

The **Details** page for the chosen Autonomous Exadata VM Cluster resource is displayed, showing at the bottom of the page 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 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 and Billing.
- CPU value of the largest provisionable Autonomous Database.
- **Provisioned CPUs** or the total CPUs allocated for all the Autonomous 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. |

| Action | Steps |
|---|---|
| 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 **Add Tag(s)**. |
| 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 *Autonomous Container Database name* 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:<br>1. Read the warning about terminating an Autonomous Container Database.<br>2. Enter the name of the Autonomous Container Database.<br>3. Click **Terminate Autonomous Container** to begin the termination process. |

# Configure and Manage Autonomous Data Guard

The Autonomous Data Guard feature of Autonomous 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 two weeks. 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.

### Required IAM Permissions

```
inspect cloud-autonomous-vmclusters

use autonomous-container-databases
```

### Procedure

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. Open the **Enable Autonomous Data Guard** dialog using any of the two options below:

   - Option 1: Under the **Autonomous Data Guard** section on the **Autonomous Container Database information** tab, click **Enable** next to **Status**.
   - Option 2: Under the **Resources** section of the **Details** page, select **Autonomous Data Guard Associations**, and click **Enable Autonomous Data Guard**.

3. Fill out the **Enable Autonomous Data Guard** 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. | |

| Setting | Description | Notes |
|---|---|---|
| Peer region | Select a region for the standby ACD. | If you choose to enable a cross-region Autonomous Data Guard association, note the following about encryption keys:<br>• If you are managing keys using the OCI Vault service, you must use a key from a virtual private vault located in the region of the primary database. This vault must be replicated in the region of the standby database. See Replicating Vaults and Keys for information on creating and managing these resources.<br>• The replicated vault used by the standby is read-only. So, when the standby assumes the primary role from a switchover or failover, you cannot create a new Pluggable Database or rotate the key. |
| 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. | |
| Protection Mode | Select **Maximum performance** or **Maximum availability** from the drop-down list. | Maximum Performance is selected by default.<br>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. |
| Automatic Failover | Select **Enable automatic failover** to enable it. You can also deselect **Enable automatic failover** if you wish to disable automatic failover for this Autonomous Data Guard setup. | |

| Setting | Description | Notes |
|---|---|---|
| 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. | Minimum fast start lag limit value supported by ACDs with 19.17 and older versions is 30 seconds.<br>For more details about automatic failover, see Automatic Failover or Fast-Start Failover. |
| Peer database backup configuration | Select a Backup Destination type from the drop-down list. | APPLIES TO: Oracle Public Cloud only<br>A backup destination is required on Exadata Cloud@Customer deployments. |
| 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. |

4. Click **Enable Autonomous Data Guard**.

> **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

1. 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 the **Autonomous Data Guard** section on the **Autonomous Container Database Information** tab.

2. You can also view the Autonomous Data Guard details in the side menu, under **Resources** by clicking **Autonomous Data Guard Associations**.

The **Autonomous Data Guard** table displays information about the peer container database, the current apply lag and transport lag, state, and last role change and creation dates.

## 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 primary or standby Autonomous Container Database 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 Autonomous Container Databases in an Autonomous Data Guard configuration where the standby is in the **snapshot standby** role.

2. In the side menu, under **Resources**, click **Autonomous Data Guard Associations**.
3. In the **Created** column of the table showing Autonomous Data Guard associations, click the ellipsis icon and then click **Switchover**.

   Oracle Autonomous Database on Dedicated Exadata Infrastructure sets the statuses of both 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 statuses of both container databases returns to **Active**.

   > **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 create a new Pluggable Database or rotate the key.

## 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 Autonomous Container Database in the Autonomous Data Guard configuration.

   For instructions, see View Details of an Autonomous Container Database.
2. In the side menu, under **Resources**, click **Autonomous Data Guard Associations**.
3. In the **Created** column of the table showing Autonomous Data Guard associations, click the ellipsis icon and then click **Failover**.
4. 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.

   Oracle Autonomous Database on Dedicated Exadata Infrastructure sets the status 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.

> **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 create a new Pluggable Database or rotate the key.

## 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.

### Required IAM Policies

```
use autonomous-container-databases
```

### Procedure

1. Go to the **Details** page of the disabled standby Autonomous Container Database that you want to reinstate.

> **Tip:**
>
> The primary database that you failed over is labeled as "Disabled Standby" in the list of Autonomous Container Databases for a compartment.

For instructions, see View Details of an Autonomous Container Database.

2. In the **Resources** section, click **Autonomous Data Guard Associations** to display a list of peer databases for the primary database you are managing.
3. For the Autonomous Container Database you are reinstating, click the ellipsis in the **Created** column, and click **Reinstate**.

   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. Under **More Actions**, click **Update Autonomous Data Guard**.

   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.

      > **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. Click **Save 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. Under **More actions**, click **Convert to snapshot standby**.

   > **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. Under **More actions**, click **Convert to physical standby**.
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.

## 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. Click the **Edit Maintenance Preferences** button. For Exadata Cloud@Customer deployments, you not locate this button, find it under **More actions**.
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 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 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** 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.

> **Tip:**
>
> For guidance on choosing a custom schedule, see Settings in Maintenance Schedule that are Customizable.

4. When finished, click **Save 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 the **View** link in the **Next maintenance** field.

   The **Autonomous Container Database Maintenance** page lists any upcoming planned maintenance.

   For more information about the details displayed in the **Autonomous Container Database Maintenance** page, see Scheduled Maintenance Details.

3. For each maintenance event listed in the **Autonomous Container Database 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 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 the **View** link in the **Next maintenance** field.

   The **Autonomous Container Database Maintenance** page lists any upcoming planned maintenance.

3. Click **Maintenance History** in the side menu to see a list of past maintenance.

# 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).

> **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 **More actions**, click the **Schedule maintenance** button.
3. Choose the following on the **Schedule maintenance** dialog that appears.

| Setting | Description | Notes |
|---|---|---|
| **Configure update type** | **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.<br>**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. |
| **Configure maintenance method** | The **Rolling** method updates the ACD, one node at a time, with no downtime for theAutonomous Databases.<br>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 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. |
| **Schedule start time** | Using the calendar tool, choose a date and time for the maintenance update. | |

4. Click **Schedule 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.

# 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 **More actions**, click **Edit Backup Settings**.

> **Tip:**
>
> You can also edit the backup settings by clicking the **Edit** link under the **Backup** section on the **Autonomous Container Database Information** tab.

3. The **Edit Backup Settings** dialog opens.

4. 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.<br>The possible options are:<br>• **Object Storage:** Stores backups in an Oracle-managed object storage container on Oracle Cloud Infrastructure.<br>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.<br>• **Network File System (NFS):** Stores backups in a Network File System (NFS) storage location.<br>If you choose Network File System (NFS) as the type, select a previously defined Backup Destination that uses Network File System (NFS) storage.<br>• **Recovery Appliance:** Stores backups in one of your previously defined backup destinations that use Oracle Zero Data Loss Recovery Appliance.<br>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 Autonomous Container Database, and the VPC user name password.<br>Provide the connection string that connects to the recovery appliance in an Oracle "easy connect" string format, that is, `<host>:<port>/<service name>`, where `<host>` is the SCAN hostname of the Zero Data Loss Recovery Appliance. | APPLIES TO: Exadata Cloud@Customer only<br>As of the current release, the backup destination type can only be set while enabling automatic backups on an ACD and can not be changed later.<br>To learn more about configuring NFS backup destination for Cloud@Customer, see Prerequisites for Backup Destinations for Exadata Cloud at Customer. |

| 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, the backup retention policy value defaults to 15 days.<br>On Exadata Cloud@Customer deployments, for Object Storage and Network File System (NFS) backup destination types, the backup retention policy value defaults to 30 days.<br>For the Recovery Appliance backup destination type, this value is controlled by the Recovery Appliance protection policy.<br>All the backups are automatically deleted after the backup retention period. |

5. When finished, click **Save Changes**.

# 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**.
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. Click **Terminate Autonomous Container** 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 both the primary and standby container databases. You cannot terminate a primary container database that contains autonomous databases.
- Terminating a **standby container database** terminates the standby container database and removes the Autonomous Data Guard configuration from the primary container database, making it a standard container databae.

# Create and Manage Autonomous Databases

## Create an Autonomous Database

You can create an Autonomous Database on Dedicated Exadata Infrastructure from the Oracle Cloud Infrastructure console.

> **Note:**
>
> In an Autonomous Data Guard setup, you can not create an Autonomous 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 Database Dedicated for Developers and Database Users Workshop.

### Procedure

1. Go to Autonomous Database in the Oracle Cloud Infrastructure Console.

   For instructions, see Access Dedicated Autonomous Database in the Oracle Cloud Infrastructure Console.
2. 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.
3. In the side menu's list of Autonomous Database resource types, click **Autonomous Database**.
4. Click **Create Autonomous Database**.
5. Fill out the **Create Autonomous Database** page with the following information:

| Setting | Description | Notes |
|---|---|---|
| **Compartment** | Select a compartment to host the Autonomous Database. | |

| Setting | Description | Notes |
|---|---|---|
| **Display name** | Enter a user-friendly description or other information that helps you easily identify the Autonomous Database. | Avoid entering confidential information. |
| **Database name** | Provide a name for the new Autonomous Database. | Must consist of letters and numbers only. The maximum length is 30 characters.<br>The same database name cannot be used for multiple Autonomous Databases in the same Autonomous Exadata VM Cluster. |
| **Workload type** | Choose **Data Warehouse** or **Transaction Processing**. | Determines the configuration of the new Autonomous Database.<br>JSON and APEX workload types are not supported on dedicated Exadata infrastructure. |
| **Deployment type** | Select **Dedicated Infrastructure**. | APPLIES TO: ✔ Oracle Public Cloud only |
| **Autonomous Container Database** | Select an Autonomous Container Database (ACD) in which to create the Autonomous Database.<br>Start by choosing whether you want a standard ACD or an ACD configured to use Autonomous Data Guard.<br>Click **Change Compartment** to select from a different compartment. | 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 Manage Primary and Standby Databases in an Autonomous Data Guard Configuration .<br>⊗ You can not create an Autonomous Database in an ACD whose standby ACD is in a snapshot standby role.<br>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 Database in such an ACD will fail because the replicated vault used by the standby is read-only.<br>⊗ You can not create an Autonomous Database for Developers instance on an Autonomous Data Guard enabled container database. |

| Setting | Description | Notes |
|---|---|---|
| **Configure the database: Free instance** | Toggle the **Free instance** button on, if you want to create an Autonomous Database for Developers instance.<br>**ECPU count** and **Storage (GB)** are auto-populated with 4 and 32 respectively because Autonomous Database for Developers comes fixed at 4 ECPUs and 32GB storage.<br>**Compute auto scaling** is disabled because developer database instances do not support manual or auto-scaling. | Autonomous Database for Developers is a free tier offering designed for database development and functional testing activities. See Autonomous Database for Developers for more details.<br>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. |
| **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 Database for more details.<br>For databases that need 1 or more OCPUs or 2 or more ECPUs, you must specify the number of assigned CPUs as an *integer*. For example, you cannot assign 3.5 ECPUs to a database. The next available number of ECPUs above 3 is 4.<br>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.<br>⊗ CPU overprovisioning is not supported for Autonomous Databases that use ECPUs. |
| **Configure the database: CPU auto scaling** | Enable or disable CPU auto scaling, which permits Autonomous Database to automatically use up to three times as many CPUs as specified by **CPU Count** as the workload on the database increases. | |

| Setting | Description | Notes |
|---|---|---|
| **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 Data Warehouse and 32 GB for the Autonomous Transaction Processing workloads. |
| **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. |
| **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.<br>Depending on the types of addresses in your list, select one of the following options from the **IP notation type** drop-down selector:<br>• **IP Address:** Specify individual IP addresses. You can use commas to separate multiple IP addresses.<br>• **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 Database by allowing only the client with specific IP addresses to connect to the database. See Restrict Access Using a Network Access Control List for details. |

| Setting | Description | Notes |
|---|---|---|
| **Standby database access control** | If the parent Autonomous Container Database uses Autonomous Data Guard, you can define access control for the standby database also.<br>Depending on your preference, select one of the following options for **Standby database access control**:<br>• 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.<br>• 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. |
| **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 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 Database.<br>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 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. Click **Create Autonomous Database**.

Your display shows **Provisioning** until the new Autonomous Database is available.

> **Note:**
>
> When you create an Autonomous Database, CPU billing is initiated, billed by the second with a minimum usage period of one minute.

# View Details of an Autonomous Database

You view the details of an Autonomous 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 you are creating a database on Oracle Cloud or Oracle Exadata Cloud@Customer:
   - **On Oracle Cloud:** click **Autonomous Data Warehouse** or **Autonomous Transaction Processing** depending on your workload type.
     The **Autonomous Databases** page opens, showing the list of databases in your current region and compartment.
   - **On Oracle Exadata Cloud@Customer:** click **Exadata Cloud@Customer**.

     The **VM Clusters** page opens. On this page's side menu, click **Autonomous Databases**.
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. In the list of autonomous databases, select the name of the database you want.

   The **Details** page for the selected database is displayed.

# Start, Stop, and Restart an Autonomous Database

You can manage start, stop, and restart an Autonomous Database from its **Details** page.

**Required IAM Policies**

```
use autonomous-databases
```

## Stop an Autonomous Database

1. Go to the **Details** page of the Autonomous 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 Database.

2. On the **Details** page, select **Actions** and then select **Stop**.
3. Click **Stop** to confirm.

> **Note:**
>
> When you stop an Autonomous 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 Database

1. Go to the **Details** page of the Autonomous 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 Database.

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

   **Start** is only shown for a stopped database.

3. Click **Start** to confirm.

> **Note:**
>
> When you start an Autonomous Database, CPU billing is initiated, billed by the second with a minimum usage period of one minute.

## Restart an Autonomous Database

1. Go to the **Details** page of the Autonomous 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 Database.

2. On the **Details** page, select **More Actions** and then select **Restart**.
3. In the confirmation dialog, select **Restart** to confirm.

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

# Enable or Disable Database In-Memory

You can enable or disable Database In-Memory from the **Details** page of an Autonomous Database.

Please see Database In-Memory for the requirements and guidelines to use this feature with Autonomous 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 Database

> **Note:**
>
> For databases that use Autonomous Data Guard, go to the **Details** page of the primary database.

2. Open the **Manage scaling** dialog using any of the two options below:
   - Option 1: Click **Manage scaling**.
   - Option 2: Under **Resource allocation**, click **Enable** next to **In-Memory Area**.
3. To enable Database In-Memory, select the **Enable database In-memory** check-box. If this box is pre-selected, deselecting it will disable Database In-Memory.
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. Click **Save**.

# Enable or Disable Auto Scaling of an Autonomous 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 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 Database on Dedicated Exadata Infrastructure.
2. For Autonomous Database on Oracle Public Cloud, select **Manage Scaling** on the **Details** page. For Autonomous Database on Exadata Cloud@Customer, select **Scale Up/Down** on the **Details** page.
3. To enable auto scaling, select the check-box next to **OCPU auto scaling** or **ECPU auto scaling**. If this check-box is pre-selected, deselecting it will disable auto scaling.

> **Tip:**
>
> The CPU type, that is, OCPU or ECPU is determined by the compute model of the parent Autonomous Exadata VM Cluster. See Compute Models in Autonomous Database for more details.

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

> **Tip:**
>
> Enter the following SQL commands to view:
>
> - The number of ECPUs the database is currently using:
>
>   ```
>   SELECT AVG_RUNNING_SESSIONS FROM V$RSRCPDBMETRIC;
>   ```
>
> - The number of OCPUs the database is currently using:
>
>   ```
>   SELECT AVG_RUNNING_SESSIONS/2 FROM V$RSRCPDBMETRIC;
>   ```

# Manage CPU or Storage Resources of an Autonomous Database

You can manage CPU or storage resources of an Autonomous Database from its **Details** page.

> **Note:**
>
> In an Autonomous Data Guard setup, you can not scale an Autonomous 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 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 Database Dedicated for Developers and Database Users Workshop.

### Procedure

1. Go to the **Details** page of the Autonomous 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 Database on Dedicated Exadata Infrastructure.

2. For Autonomous Database on Oracle Public Cloud, select **Manage Scaling** on the **Details** page. For Autonomous Database on Exadata Cloud@Customer, select **Scale Up/Down** on the **Details** page.

   ⊗ This option is not enabled for an Autonomous Database for Developers instance.

3. Select the change in resources for your scale request:

   - Click the up arrow to select a value for **OCPU count** or **ECPU 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 Transaction Processing workloads and to the **low** services for the Autonomous Data Warehouse workloads. See Predefined Database Service Names for Autonomous Databases to view a list of predefined services supported by an Autonomous 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 Database

### Required IAM Policies

```
use autonomous-databases
```

### Procedure

1. Go to the **Details** page of the Autonomous 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 Database on Dedicated Exadata Infrastructure.

2. On the **Details** page, select **Manage Scaling**.

   ⊗ This option is not enabled for an Autonomous Database for Developers instance.

3. On the **Manage Scaling** page, select the change in resources for your scale request:

   - Click the down arrow to select a value for **OCPU count** or **ECPU 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 Databases to view a list of predefined services supported by an Autonomous 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.

# Move an Autonomous Database to a Different Compartment

Shows you the steps to move an Autonomous Database to a different Oracle Cloud Infrastructure compartment.

> **Note:**
>
> As soon as you move an Autonomous 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 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 Database you want to move.

   For instructions, see View Details of a Dedicated Autonomous Database.
2. On the **Details** page, select **More Actions** and then select **Move Resource**.
3. In the **Move Resource to a Different Compartment** page, select the new compartment.
4. Click **Move Resource**.

See Moving Database Resources to a Different Compartment for more information.

# Clone an Autonomous Database on Dedicated Exadata Infrastructure

Cloning is the process of creating a point-in-time copy of your Autonomous Database or its backup set. You can clone an Autonomous Database on Dedicated Exadata Infrastructure from its **Details** page.

# Clone an Autonomous Database Instance

### Prerequisites

To view the requirements for successfully cloning an Autonomous Database instance, see Clone Requirements.

### Required IAM Policies

```
manage autonomous-databases

read autonomous-container-databases
```

### Procedure

Follow these steps to clone an Autonomous Database instance using the Oracle Cloud Infrastructure console.

1. Go to the **Details** page of the Autonomous 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 Database.
2. On the **Details** page, select **More Actions** and then select **Create Clone**.

   **Create Clone** is only enabled only when the database's state is **Available**.
3. Fill out the **Create Autonomous 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 Database clone: Compartment** | Choose a compartment for the database clone. | See Understanding Compartments for information on using and managing compartments. |
| **Autonomous Database clone: Source database name** | This read-only field identifies the name of the source database. | |
| **Autonomous 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 Database clone: Database name** | Specify a name for the clone database. | Database name must consist of letters and numbers only.<br>The maximum length is 30 characters.<br>The same database name cannot be used for multiple Autonomous 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. |
| **Exadata Infrastructure** | Specify the Exadata Infrastructure to place the clone database. | The default is the same Exadata Infrastructure as that of the source database.<br>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. | The default is the same AVMC resource as that of the source database.<br>You can choose an AVMC resource from a different compartment by clicking **Change Compartment**, if necessary. |
| **Autonomous Container Database** | Specify the Autonomous Container Database (ACD) in which to place the cloned database.<br>Start by choosing whether you want a standard ACD or an ACD configured to use Autonomous Data Guard.<br>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 Database for Developers instance.<br>The **ECPU count** and **Storage (GB)** are auto-populated with 4 and 32, respectively, because Autonomous Database for Developers instances come fixed at 4 ECPUs and 32GB of storage.<br>**Compute auto-scaling** is disabled because developer database instances do not support manual or auto-scaling. | Autonomous Database for Developers is a free tier offering designed for database development and functional testing activities. For more details, see Autonomous Database for Developers.<br>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.<br>To successfully clone an Autonomous Database to an Autonomous Database for Developers, the actual used space of the source database, rounded up to the next GB, must be *32GB or less*. |
| **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.<br>⊗ You can clone an OCPU database into either OCPU or ECPU database. However, you can not clone an ECPU database into an OCPU database.<br>This value defaults to 2 ECPUs or 1 OCPU depending on the CPU type. Refer to Compute Models in Autonomous Database for more details.<br>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. |

| Setting | Description | Notes |
|---|---|---|
| **Configure the database: Auto scaling** | Enable or disable auto scaling, which permits Autonomous Database to automatically use up to three times the allocated CPUs as the workload on the database increases. | |
| **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. |
| **Administrator credentials** | **Username:** This is a read-only field auto-populated as ADMIN.<br>**Password:** Set the password for the ADMIN database user in the cloned database.<br>**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.<br>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 Database for details. |
| **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 apply from the target Autonomous Container Database. | The source and target Autonomous Container Database's encryption key settings have to match for the clone to be successful. |
| **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 Database.<br>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. |

| 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 Dedicated Autonomous Database Cloud Resources. |

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

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

## Clone Autonomous Database from Backup

### Prerequisites

To view the requirements for successfully cloning an Autonomous Database instance, see Clone Requirements.

### Required IAM Policies

```
manage autonomous-databases

read autonomous-container-databases
```

### Procedure

Follow these steps to clone an Autonomous Database from backup using the Oracle Cloud Infrastructure console.

1. Go to the **Details** page of the Autonomous Database you want to clone.

   For instructions, see View Details of a Dedicated Autonomous Database.

2. There are two ways to open the **Create Autonomous Database Clone** page from the **Details** page:

   - **Option 1:** Select **More Actions** and then select **Create Clone**.

     When you open the **Create Autonomous 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:** Scroll down to the bottom of the **Details** page, and click **Backups** under **Resources**. Then, click ⋮, and click **Create Clone** for the backup to be cloned.

     When you open the **Create Autonomous 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 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 Database backup.<br>See Clone Types for more details. |
| **Clone source** | Select **Clone from a backup**. | When you open the **Create Autonomous 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. |
| **Choose a specific backup** | Once you select **Clone from a backup**, you see two additional options to choose a specific backup.<br>**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 value within the last 60 days.<br>**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. | **Point in time clone** is the default option. You can not use this option to clone a long-term backup.<br>You must choose **Select the backup from a list** option if you want to clone from a long-term backup. You can only select *one* 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. |

| Setting | Description | Notes |
|---|---|---|
| **Database name** | Specify a name for the clone database. | Database name must consist of letters and numbers only.<br>The maximum length is 30 characters.<br>The same database name cannot be used for multiple Autonomous 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. |
| **Exadata Infrastructure** | Specify the Exadata Infrastructure to place the clone database. | The default is the same Exadata Infrastructure as that of the source database.<br>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. | The default is the same AVMC resource as that of the source database.<br>You can choose an AVMC resource from a different compartment by clicking **Change Compartment**, if necessary. |
| **Autonomous Container Database** | Specify the Autonomous Container Database (ACD) in which to place the cloned database.<br>Start by choosing whether you want a standard ACD or an ACD configured to use Autonomous Data Guard.<br>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 Database for Developers instance.<br>The **ECPU count** and **Storage (GB)** are auto-populated with 4 and 32, respectively, because Autonomous Database for Developers instances come fixed at 4 ECPUs and 32GB of storage.<br>**Compute auto-scaling** is disabled because developer database instances do not support manual or auto-scaling. | Autonomous Database for Developers is a free tier offering designed for database development and functional testing activities. For more details, see Autonomous Database for Developers.<br>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.<br>To successfully clone a backup to an Autonomous 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 *database instance* to a developer database instead, provided that its *actual used space*, 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.<br>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 Database clone from Backup.<br>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 Database for more details.<br>⊗ You can clone an OCPU database into either OCPU or ECPU database. However, you can not clone an ECPU database into an OCPU database.<br>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 Databases.<br>The time taken to clone an Autonomous Database depends on the CPU Count and the network bandwidth between the Backup Destination and the target Autonomous Container Database.<br>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. |
| **Configure the database: Auto scaling** | Enable or disable auto scaling, which permits Autonomous Database to automatically use up to three times the allocated CPUs as the workload on the database increases. | |

| Setting | Description | Notes |
|---|---|---|
| **Administrator credentials** | **Username:** This is a read-only field auto-populated as ADMIN.<br>**Password:** Set the password for the ADMIN database user in the cloned database.<br>**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.<br>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 Database for details. |
| **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 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. |
| **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 Database.<br>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 Database Cloud Resources. |

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

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

# Managing Primary and Standby Databases in an Autonomous Data Guard Configuration

Learn how to mange primary and standby databases in an autonomous data guard configuration.

When you create an Autonomous 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 Database.

> **Note:**
>
> Because two autonomous databases are created when you use Autonomous Data Guard, twice the number of OCPU 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 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 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 OCPU 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 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 choosing **Autonomous Data Guard Assocations** under Resources in the side menu.

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 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.

## Backup an Autonomous Database on Dedicated Exadata Infrastructure Manually

In addition to automatic backups, Autonomous Database on Dedicated Exadata Infrastructure also allows you take manual backups from its **Details** page.

#### Required IAM Policies

```
read autonomous-databases

manage autonomous-backups
```

**Procedure**

1. Go to the **Details** page of the Autonomous Database you want to manually backup.

   For instructions, see View Details of an Autonomous Database on Dedicated Exadata Infrastructure.
2. On the **Details** page, under **Backups**, click **Create Manual Backup**.
3. In the **Create Manual Backup** dialog enter a name in the **Name** field.
4. In the **Create Manual Backup** dialog click **Create**.

> **Note:**
>
> 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.

## Create a Long-Term Backup

In addition to automatic backups, Autonomous 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 Database for Developers instances. See Autonomous Database for Developers for more details.

## Procedure

You can create a long-term backup from the Details page of an Autonomous 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 Database for which you want to create a long-term backup.

   For instructions, see View Details of an Autonomous 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.

2. Scroll down and locate **Backups** under **Resources**.
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 (-).<br>Must start with a letter or underscore(_), cannot contain two successive hyphens(--) or cannot end with a hyphen(-). |
| **Backup Destination** | **Backup destination type:** This is defaulted to Network File System (NFS) and can not be modified.<br>**Backup destination:** Select a backup destination from the list of backup destinations, changing compartments if necessary. | APPLIES TO: Exadata Cloud@Customer only<br>Irrespective of the parent container database's backup destination type, your long-term backup's destination remains NFS.<br>To learn more about configuring NFS backup destination for Cloud@Customer, see Prerequisites for Backup Destinations for Exadata Database Service on Cloud@Customer |
| **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.

# Manage Long-Term Backups

You can view the list of long-term backups for an Autonomous 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 Database for which you want to modify a long-term backup.

   For instructions, see View Details of an Autonomous Database on Dedicated Exadata Infrastructure.

2. Scroll down and locate **Backups** under **Resources**.

   All the existing backups for this Autonomous 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**.<br>The **Clone Autonomous Database** page opens with **Clone Source** defaulted to this long-term backup.<br>Provide the remaining details on this page, and click **Clone Autonomous Database**.<br>Refer to Clone Autonomous Database from Backup<br>for more details. |
| Edit the retention period | Click the **actions** icon, and select **Edit retention period**.<br>Choose the **Retention period** of your choice from the **Edit long-term backup retention period** dialog, and click **Save**.<br>The long-term backup's **Expiration date** updates accordingly. |

| Action | Steps |
| --- | --- |
| Delete a long-term backup | Click the **actions** icon, and select **Delete**.<br>Click **Delete** on the confirmation dialog that appears. The backups State changes to **Deleted**. |

# Restore and Recover Autonomous Database on Dedicated Exadata Infrastructure

You can restore and recover an Autonomous 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 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 Database on Dedicated Exadata Infrastructure.
2. On the **Details** page, select **More Actions** and then select **Restore**.
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.

4. Click **Restore**.

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

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

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

> **Note:**
>
> - 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.

# View Autonomous Database Metrics for a Database

Shows the steps to view the Autonomous Database metrics.

**Required IAM Policies**

- `read metrics`
- `inspect work-requests`

See Prerequisites for Monitoring Autonomous Database Metrics for more details.

## Procedure

1. Go to the **Details** page of the Autonomous Database whose metrics you want to view.

   For instructions, see View Details of a Dedicated Autonomous Database.

2. On the **Details** page, under **Resources**, select **Metrics**.
3. 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 Database Metrics for a Database

Shows the steps to view metrics for Autonomous Database in a compartment.

**Required IAM Policies**

- `read metrics`
- `inspect work-requests`

See Prerequisites for Monitoring Autonomous Database Metrics for more details.

## Procedure

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 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 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 Databases in the compartment you can show metrics aggregated across the Autonomous 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.

# Enable Operations Insights for an Autonomous Database

APPLIES TO: Oracle Public Cloud only

Autonomous Database includes Operations Insights as one of its associated services. Operations Insights is a cloud-native service that enables users to make informed, data-driven, Autonomous 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, Operations Insights is disabled for an Autonomous Database, and you must enable it from the Oracle Cloud Infrastructure console.
To enable Operations Insights for your database:

1. Go to the **Details** page of the Autonomous Database.

   For instructions, see View Details of a Dedicated Autonomous Database.

   > **Note:**
   >
   > For dedicated databases that use Autonomous Data Guard, go to the **Details** page of the primary database. Operations Insights is not supported on a Standby Autonomous Database.

2. On the **Details** page, locate **Operations Insights** under **Associated Services**.
3. Click the **Enable** link next to Operations Insights.

   > **Note:**
   >
   > For an Autonomous Database with Operations Insights already enabled, you would see the **Disable** link instead. Click this link to disable Operations Insights for your database.

4. On the confirmation dialog, click **Enable**.

You see **Enabling** next to Operations Insights. After the Operations Insights is enabled, a link to view the operations insights for your database appears. Refer to Use Operations Insights on Dedicated Autonomous Database for more details.

## Terminate an Autonomous Database on Dedicated Exadata Infrastructure

Shows you the steps to terminate an Autonomous Database on Dedicated Exadata Infrastructure.

> **Note:**
>
> Terminating an Autonomous Database permanently deletes it and removes all automatic backups and long-term backups. You cannot recover a terminated database.

**Required IAM Policies**

```
manage autonomous-databases
```

**Procedure**

1. Go to the **Details** page of the Autonomous 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 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 Database.
2. On the **Details** page, select **More Actions** and then select **Terminate**.
3. On the **Terminate Database** page enter the database name to confirm that you want to terminate the database.
4. Click **Terminate Database**.

# Connect to Autonomous Database

## Download Client Credentials

Oracle client credentials (wallet files) are downloaded from Autonomous Database by a service administrator. If you are not an Autonomous 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 Database details page.
2. Click **Database connection**.
3. On the **Database 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 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 Database.

# View Connection Strings for an Autonomous Database

You can view the TNS names and connection strings of an Autonomous 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 Database:

1. Go to the **Details** page of the Autonomous Database for which you want to view connection strings.

   For instructions, see View Details of a Dedicated Autonomous Database.

   > **Note:**
   >
   > For databases that use Autonomous Data Guard, go to the **Details** page of the primary database.

2. Click **Database connection** to open the **Database connection** dialog.
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. In the **Connection string** column, click **Show** to display the complete connection string or click **Copy** to copy the entire connection string.
5. For cross-region standby Autonomous 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.

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

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

1. Download and install the Oracle Instant Client basic package for your system's OS and architecture:
   - **Oracle Linux:**
     a. Enable the Instant Client channel:

        ```
        yum -y install oracle-release-el7
        ```

     b. Download and install the Instant Client 19.3 basic package:

        ```
        yum -y install oracle-instantclient19.3-basic
        ```

   - **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. See Download Client Credentials for information on downloading client credentials for Autonomous Database.

3. After downloading the zip file, follow these steps:

   a. Unzip the client credentials zip file.

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

   ```
   (DIRECTORY="?/network/admin")
   ```

   to:

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

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

## Prepare for JDBC Thin Connections

You prepare for JDBC thin connections by downloading and installing Oracle Database JDBC Driver and then downloading the client credentials for your Autonomous 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 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.

# 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 Database Dedicated Workshop.

Oracle Database Actions, a browser-based application that provides many database development, management and monitoring features is built into Autonomous Databases. Therefore, you can use it without downloading or installing additional software on your system. Database Actions was earlier known as SQL Developer Web.

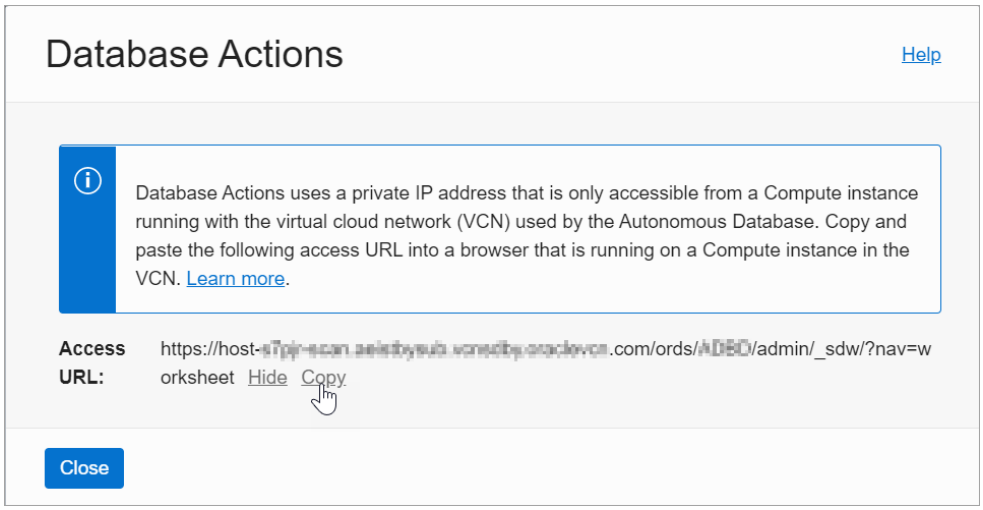
By default, only the ADMIN database user has access to Database Actions. Before you can sign in as another database user, the ADMIN user must enable access for that user, as described in Enabling User Access to Database Actions in *Using Oracle Database Actions*.

> **Note:**
>
> To sign into Database Actions, the system you are using must have network access to the Autonomous Database, as described in Configure a Development System to Access the Database.

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 Database.

   For instructions, see View Details of a Dedicated Autonomous Database.
2. Open the **Tools** tab.
3. Click **Open Database Actions** in the **Database Actions** tile.
4. In the **Database Actions** dialog, click the **Copy** link in the **Access URL** field to copy the URL to your clipboard.




5. 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 to Autonomous 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.

> **Note:**
>
> To connect Oracle SQL Developer to an Autonomous Database on Dedicated Exadata Infrastructure, the system running Oracle SQL Developer must have network access to the database, as described in Configure a Development System to Access the Database.

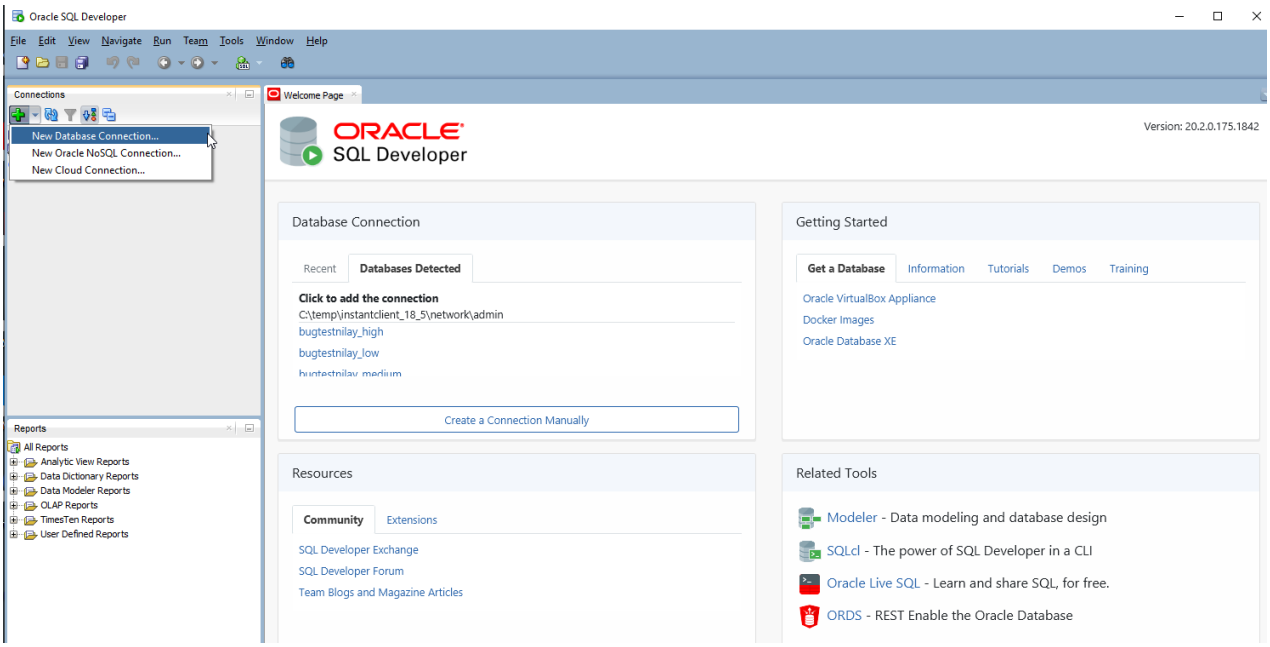
## Connect with Oracle SQL Developer Version 18.2 or Later

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

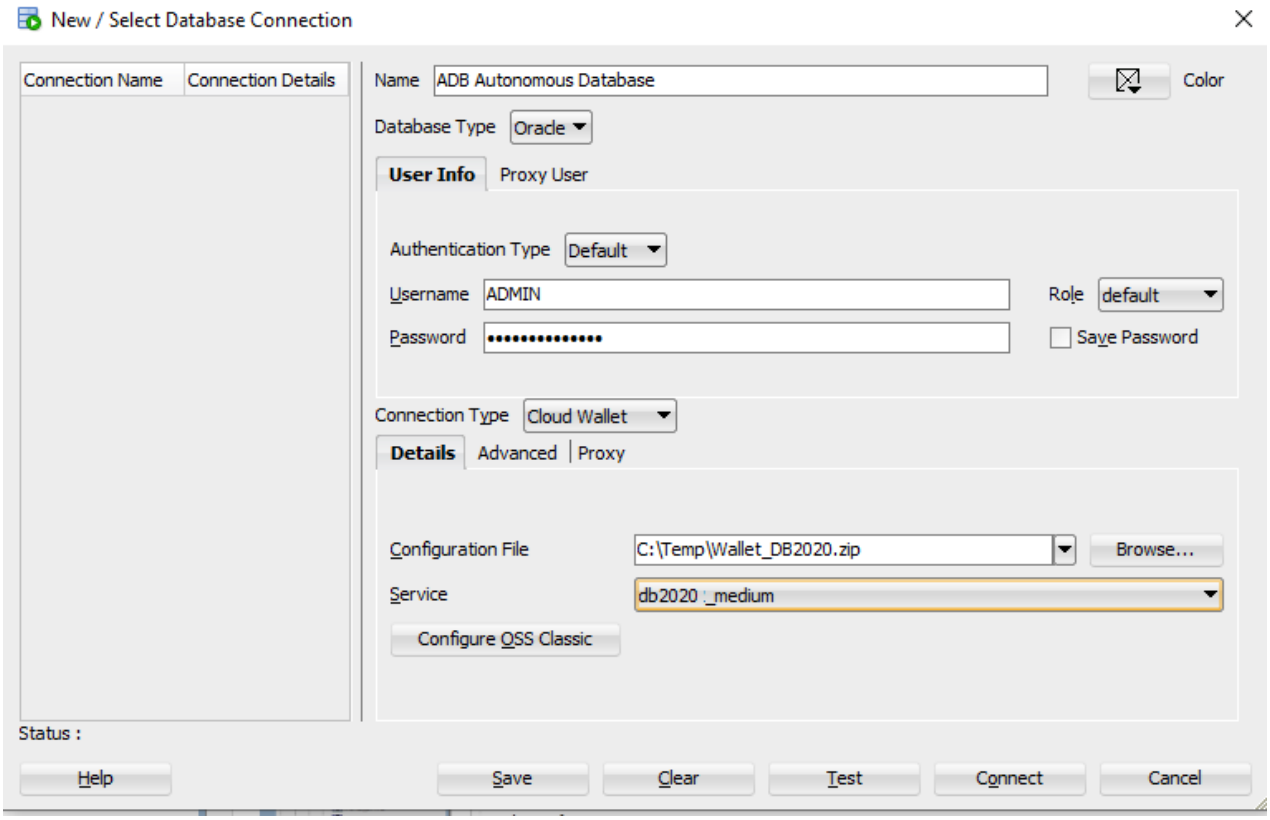
1. Download the zip file containing client credentials for your database to a secure directory on system.

   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

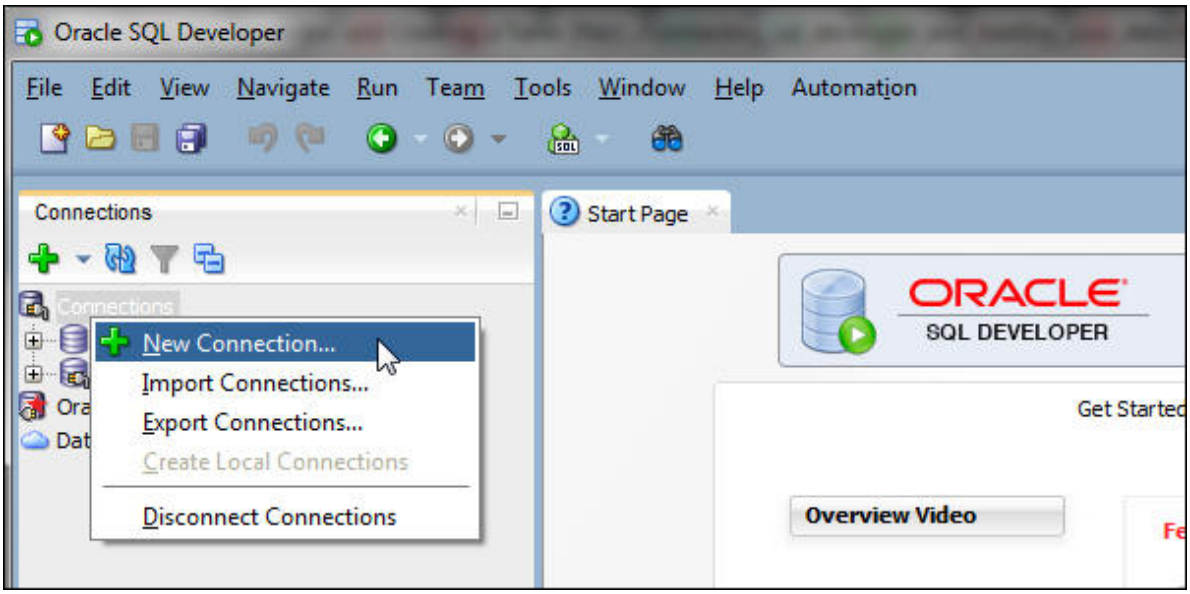
Oracle recommends that you use Oracle SQL Developer version 18.2 or later; however, earlier versions of SQL Developer will work with Autonomous Database.

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

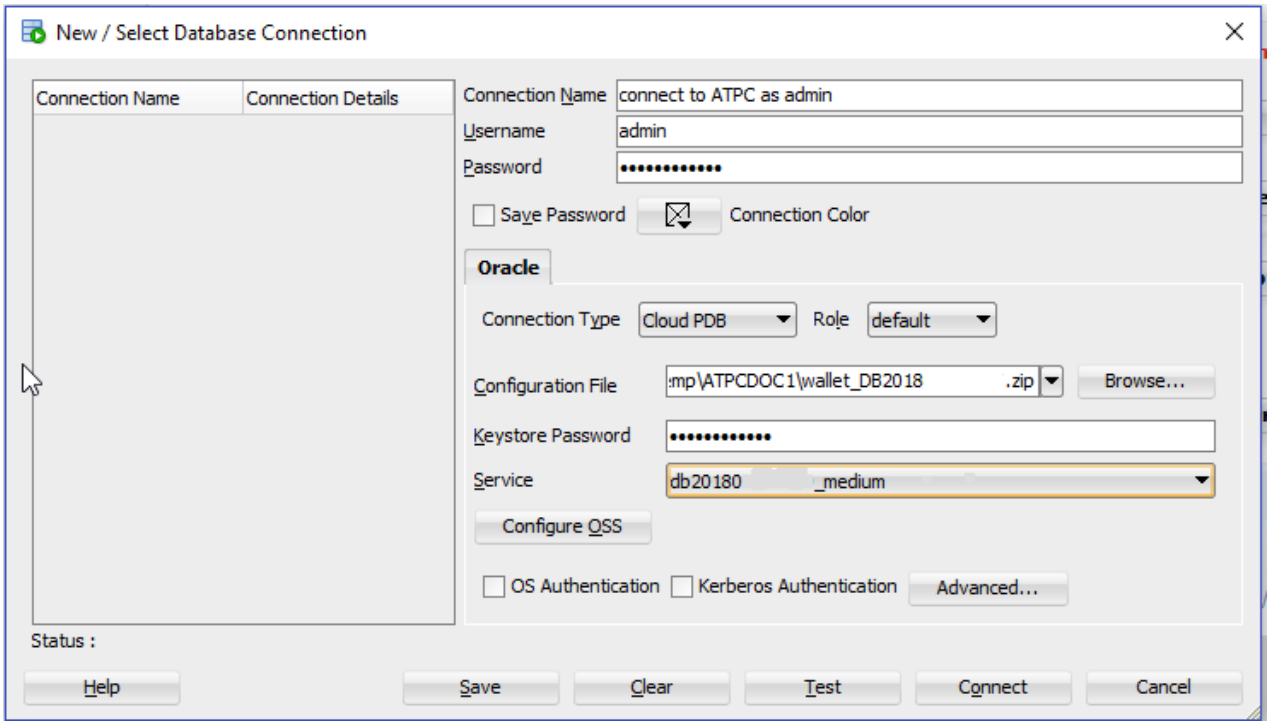
1. Download the zip file containing client credentials for your database to a secure directory on system.

   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 Database.
   - **Service**: Enter the service name. The client credentials file provides the service names.

## 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 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.

> **Note:**
>
> To connect Oracle SQLcl to an Autonomous Database, the system running Oracle SQLcl must have network access to the database, as described in Configure a Development System to Access the Database.

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

SQLcl can connect to an Autonomous 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 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 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 Databases.

For information on SQLcl, see Oracle SQLcl.

## 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 Database Dedicated Workshop.

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

> **Note:**
>
> To connect SQL*Plus to an Autonomous Database, the system running SQL*Plus must have network access to the database, as described in Configure a Development System to Access the Database.

To install and configure the client and connect to an Autonomous 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 Python Applications Using mTLS

APPLIES TO: Exadata Cloud@Customer only

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

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

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

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

2. Install the python-oracledb driver from PyPI.

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

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

   Run the following command to upgrade python:

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

   You should see output similar to the following:

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

   Notes for installing python-oracledb:

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

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

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

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

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

See Installing python-oracledb for additional options and tips.

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

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

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

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

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

   - **Mutual TLS (mTLS) Connections**:
     - If your database is on a remote computer, then download the free Oracle Instant Client “Basic” or “Basic Light” package for your operating system architecture. Use a supported version: **Oracle Instant Client**: 18.19 (or later), 19.2 (or later), or 21 (base release or later).
     - Alternatively, you can use the Full Oracle Database client libraries when they are available on your system (including Full Oracle Database Client: **Oracle Database Client**: 18.19 (or later), 19.2 (or later), or 21 (base release or later).
   - **TLS Connections**: Oracle Call Interface (OCI) clients support TLS authentication 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 Database instance.

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

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

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

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

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

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

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

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

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

To connect in Thin mode:

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

- Linux
- Windows

### Linux

For example on Linux:

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

### Windows

For example on Windows:

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

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

- Linux
- Windows

### Linux

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

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

### Windows

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

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

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

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

---

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

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

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

For example, on Linux:

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

For example, on Windows:

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

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

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

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

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

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

To connect in Thick mode:

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

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

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

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

     or

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

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

   - Alternatively, move the files to any accessible directory.

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

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

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

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

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

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

   - `config_dir`: Specifies the configuration directory when you are putting the configuration files. This is only required when the configuration files are placed in a directory outside of the instant client configuration directory `network/admin`.
   - `dsn`: Specifies the desired network alias from the `tnsnames.ora` file.
   - `password`: Specifies the database user password.
   - `user`: Specifies the database user.

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

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

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

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

- Linux
- Windows

## Linux

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

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

```
      password=password,
      dsn="db2024_low")
```

### Windows

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

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

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

---

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

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

```
SQLNET.USE_HTTPS_PROXY=on
```

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

For example:

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

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

## 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 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 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 Database instance. You obtain a zip file containing client security credentials and network configuration settings required to access your Autonomous 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 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.

# Secure Autonomous Database

## Rotate Encryption Keys

This article describes the prerequisite tasks for using customer-managed keys with Autonomous Database on Dedicated Exadata Infrastructure.

## 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. Click **Rotate Encryption Key**.
3. In the **Rotate Encryption Key** dialog, confirm that you want to rotate the key by clicking **Rotate 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**: Oracle Autonomous 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**: Oracle Autonomous Database uses the underlying technology (Oracle Cloud Infrastructure Vault for container databases on Oracle Cloud or Oracle Key Vault for container databases on Exadata Cloud@Customer) 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.
  You can view the latest Key Version OCID and the entire Key History from your Autonomous Container Database details page.

> **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 Database

You rotate the encryption key of an Autonomous Database from its **Details** page.

1. Go to the **Details** page of the Autonomous Database whose encryption key you want to rotate.
   For instructions, see View Details of a Dedicated Autonomous Database.
2. Click **More Actions** and then click **Rotate Encryption Key**.
3. In the **Rotate Encryption Key** dialog, confirm that you want to rotate the key by clicking **Rotate Key**.

   The Autonomous Database goes to the **Updating** status, the encryption key is rotated, and the Autonomous 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**: Oracle Autonomous Database rotates the encryption key, storing the new value in the secure key store on the Exadata system where the Autonomous Database resides.
   - **Customer-managed key**: Oracle Autonomous Database uses the underlying technology (Oracle Cloud Infrastructure Vault for databases on Oracle Cloud or Oracle Key Vault for databases on Exadata Cloud@Customer) 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 Database.
     You can view the latest Key Version OCID and the entire Key History from your Autonomous Database details page.

> **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.

# Register or Deregister a 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 Database, run the *labs 2 to 13* in Oracle Autonomous 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 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 Database.

**Procedure**

1. Go to the **Details** page of the Autonomous Database you want to register or deregister with Oracle Data Safe.

   For instructions, see View Details of a Dedicated Autonomous 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.

# Set Access Control List for an Autonomous Database

An Access Control List (ACL) provides additional protection to your Autonomous 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 Database to which you want to set access control.

   For instructions, see View Details of a Dedicated Autonomous Database.

2. In the **Autonomous 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 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 Changes**.

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.

7

# 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 Databases deployed on Oracle Public Cloud 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
- Exadata Cloud@Customer

#### Oracle Public Cloud

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 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 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.

## 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 Database. |
| DROP_CREDENTIAL Procedure | This procedure removes an existing credential from Autonomous Database. |
| UPDATE_CREDENTIAL Procedure | This procedure updates cloud service credential attributes in Autonomous Database. |

# CREATE_CREDENTIAL Procedure

This procedure stores cloud service credentials in Autonomous Database.

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 |
|---|---|
| `credential_name` | The name of the credential to be stored. The `credential_name` parameter must conform to Oracle object naming conventions, which do not allow spaces or hyphens. |
| `username` | The `username` and `password` arguments together specify your cloud service credentials. See the usage notes for what to specify for the `username` and `password` for different cloud services. |
| `password` | The `username` and `password` arguments together specify your cloud service credentials. |
| `user_ocid` | Specifies the user's OCID. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the User's OCID. |
| `tenancy_ocid` | Specifies the tenancy's OCID. See Where to Get the Tenancy's OCID and User's OCID for details on obtaining the Tenancy's OCID. |
| `private_key` | Specifies the generated private key. Private keys generated with a passphrase are not supported. You need to generate the private key without a passphrase. See How to Generate an API Signing Key for details on generating a key pair in PEM format. |
| `fingerprint` | Specifies a fingerprint. After a generated public key is uploaded to the user's account the fingerprint is displayed in the console. Use the displayed fingerprint for this argument. See How to Get the Key's Fingerprint and How to Generate an API Signing Key for more details. |

### 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 . 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..aaaaaaaauq54mi7zdyfhw33ozkwuontjceel7fok5nq3bf2vwetkpq
soa',
       tenancy_ocid    =>
'ocid1.tenancy.oc1..aabbbbbbaafcue47pqmrf4vigneebgbcmmoy5r7xvoypicjqqge
32ewnrcyx2a',
       private_key     =>
```

```
'MIIEogIBAAKCAQEAtUnxbmrekwgVac6FdWeRzoXvIpA9+0r1.....wtnNpESQQQ0QLGPD8NM//
JEBg=',
       fingerprint     => 'f2:db:f9:18:a4:aa:fc:94:f4:f6:6c:39:96:16:aa:27');
END;
/
```

Private keys generated with a passphrase are not supported. You need to generate the private key without a passphrase. See How to Generate an API Signing Key for more information.

#### 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.<br>See Projects and HMAC Keys for more information. |

## DROP_CREDENTIAL Procedure

This procedure removes an existing credential from Autonomous Database.

#### Syntax

```
DBMS_CLOUD.DROP_CREDENTIAL (
   credential_name      IN VARCHAR2);
```

#### Parameters

| Parameter | Description |
|---|---|
| `credential_name` | The name of the credential to be removed. |

## 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.<br>For a username/password type credential, the valid attribute values are: USERNAME and PASSWORD.<br>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 Databases deployed on Oracle Public Cloud 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
- Exadata Cloud@Customer

### Oracle Public Cloud

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 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 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.

## 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 Database tables either from Cloud Object Storage or from files in a directory. |
| COPY_DATA Procedure for Avro, ORC, or Parquet Files | This procedure with the `format` parameter `type` set to the value `orc`, `parquet`, or `avro` loads data into existing Autonomous Database tables from ORC, Parquet, or Avro files in the Cloud or from ORC, Parquet, or Avro files in a directory.<br>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. |

| Subprogram | Description |
|---|---|
| CREATE_EXTERNAL_TABLE Procedure | This procedure creates an external table on files in the Cloud or on files in a directory. This allows you to run queries on external data from Autonomous Database. |
| CREATE_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.<br>This allows you to run queries on external data from Autonomous Database. |
| CREATE_HYBRID_PART_TABLE Procedure | This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous Database. |
| DELETE_ALL_OPERATIONS Procedure | This procedure clears either all data load operations logged in the `user_load_operations` table in your schema or clears all the data load operations of the specified type, as indicated with the `type` parameter. |
| DELETE_FILE Procedure | This procedure removes the specified file from the specified directory on Autonomous Database |
| DELETE_OBJECT Procedure | This procedure deletes the specified object on object store. |
| EXPORT_DATA Procedure | This procedure exports data from Autonomous Database to files in the Cloud based on the result of a query. The overloaded form enables you to use the `operation_id` parameter. Depending on the `format` parameter `type` option specified, the procedure exports rows to the Cloud Object store as text with options of CSV, JSON, 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 Database. The function form reads an object from Cloud Object Storage and returns a `BLOB` to Autonomous Database. |
| LIST_FILES Function | This function lists the files in the specified directory. The results include the file names and additional metadata about the files such as file size in bytes, creation timestamp, and the last modification timestamp. |
| LIST_OBJECTS Function | This function lists objects in the specified location on object store. The results include the object names and additional metadata about the objects such as size, checksum, creation timestamp, and the last modification timestamp. |
| MOVE_OBJECT Procedure | This procedure moves an object from one Cloud Object Storage bucket to another one. |
| PUT_OBJECT Procedure | This procedure is overloaded. In one form the procedure copies a file from Autonomous Database to the Cloud Object Storage. In another form the procedure copies a `BLOB` from Autonomous Database to the Cloud Object Storage. |
| SYNC_EXTERNAL_PART_TABLE Procedure | This procedure simplifies updating an external partitioned table from files in the Cloud. Run this procedure whenever new partitions are added or when partitions are removed from the Object Store source for the external partitioned table. |

| Subprogram | Description |
|---|---|
| VALIDATE_EXTERNAL_TABLE Procedure | This procedure validates the source files for an external table, generates log information, and stores the rows that do not match the format options specified for the external table in a *badfile* table on Autonomous Database. |
| VALIDATE_EXTERNAL_PART_TABLE Procedure | This procedure validates the source files for an external partitioned table, generates log information, and stores the rows that do not match the format options specified for the external table in a *badfile* table on Autonomous Database. |
| VALIDATE_HYBRID_PART_TABLE Procedure | This procedure validates the source files for a hybrid partitioned table, generates log information, and stores the rows that do not match the format options specified for the hybrid table in a *badfile* table on Autonomous Database. |

## COPY_COLLECTION Procedure

This procedure loads data into a SODA collection from Cloud Object Storage or from a directory. If the specified SODA collection does not exist, the procedure creates it. The overloaded form enables you to use the `operation_id` parameter.

### 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.<br>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.<br>**Cloud source file URIs**<br>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.<br>The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.<br>**Directory**<br>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.<br>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?'<br>To specify multiple directories, use a comma separated list of directories: For example:'*MY_DIR1*:*, *MY_DIR2*:test?'<br>Use double quotes to specify a case-sensitive directory name. For example:'"*my_dir1*":*, "*my_dir2*":Test?'<br>To include a quote character, use two quotes. For example:'*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote (). |
| format | The options describing the format of the source files. These options are specified as a JSON string. Supported formats for JSON data are: characterset, compression, ignoreblanklines, jsonpath, maxdocsize, recorddelimiter, rejectlimit, unpackarray.<br>Apart from the mentioned formats for JSON data, Autonomous Database supports other formats too. For the list of format arguments supported by Autonomous Database, see Format Parameter. |
| operation_id | Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view. |

### Example

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(
            credential_name => 'OBJ_STORE_CRED',
            username        => 'user_name@oracle.com',
            password        => 'password'
            );

    DBMS_CLOUD.COPY_COLLECTION(
            collection_name => 'myCollection',
            credential_name => 'OBJ_STORE_CRED',
```

```
            file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/adbexample/b/json/o/myCollection.json'
            );
END;
/
```

## COPY_DATA Procedure

This procedure loads data into existing Autonomous Database tables from files in the Cloud, or from files in a directory. The overloaded form enables you to use the `operation_id` parameter.

### Syntax

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

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

### Parameters

| Parameter | Description |
|---|---|
| `table_name` | The name of the target table on the database. The target table needs to be created before you run `COPY_DATA`. |
| `credential_name` | The name of the credential to access the Cloud Object Storage.<br>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.<br>**Cloud source file URIs**<br>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.<br>The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.<br>**Directory**<br>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.<br>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?'<br>To specify multiple directories, use a comma separated list of directories: For example:'*MY_DIR1*:*, *MY_DIR2*:test?'<br>Use double quotes to specify a case-sensitive directory name. For example:'"*my_dir1*":*, "*my_dir2*":Test?'<br>To include a quote character, use two quotes. For example:'*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote (').<br>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. |
| schema_name | |
| field_list | Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the target table definition. This argument's syntax is the same as the field_list clause in regular Oracle external tables. For more information about field_list see *Oracle® Database Utilities*.<br>When the format parameter type option value is json, this parameter is ignored.<br>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.<br>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 `recorddelmiter` format option.

## COPY_DATA Procedure for Avro, ORC, or Parquet Files

This procedure with the `format` parameter `type` set to the value `avro`, `orc`, or `parquet` loads data into existing Autonomous Database tables from Avro, ORC, or Parquet files in the Cloud 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.<br>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.<br>**Cloud source file URIs**<br>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.<br>The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.<br>**Directory**<br>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.<br>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?'<br>To specify multiple directories, use a comma separated list of directories: For example:'*MY_DIR1*:*, *MY_DIR2*:test?'<br>Use double quotes to specify a case-sensitive directory name. For example:'"*my_dir1*":*, "*my_dir2*":Test?'<br>To include a quote character, use two quotes. For example:'*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote (').<br>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. |
| schema_name | |
| field_list | Ignored for Avro, ORC, or Parquet files.<br>The fields in the source match the external table columns by name. Source data types are converted to the external table column data type.<br>For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping.<br>For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details on mapping.<br>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.<br>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.<br>This parameter is mandatory.<br>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.<br>This parameter is mandatory.<br>The format of the URIs depends on the Cloud Object Storage service. See Cloud Object Storage URI Formats for more information. |

| Parameter | Description |
|---|---|
| target_credential_name | The name of the credential to access the target Cloud Object Storage location.<br>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_TABLE Procedure

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

### Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       IN VARCHAR2,
    credential_name  IN VARCHAR2,
    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.<br>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.<br>**Cloud source file URIs**<br>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.<br>The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.<br>**Directory**<br>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.<br>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?'<br>To specify multiple directories, use a comma separated list of directories: For example: '*MY_DIR1*:*, *MY_DIR2*:test?'<br>Use double quotes to specify a case-sensitive directory name. For example: '"*my_dir1*":*, "*my_dir2*":Test?'<br>To include a quote character, use two quotes. For example: '*MY_DIR*:''filename.ext'. This specifies the filename starts with a quote (') . |
| column_list | Comma-delimited list of column names and data types for the external table. |
| field_list | Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the *column_list* parameter. This argument's syntax is the same as the field_list clause in regular Oracle external tables. For more information about field_list see *Oracle® Database Utilities*. |
| format | The options describing the format of the source files. For the list of the options and how to specify the values see Format Parameter.<br>For Avro, ORC, or Parquet format files, see CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files. |

### Usage Notes

- The procedure DBMS_CLOUD.CREATE_EXTERNAL_TABLE supports external partitioned files in the supported cloud object storage services, including:
  - Oracle Cloud Infrastructure Object Storage
  - Azure Blob Storage
  - Amazon S3
  - Google Cloud 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 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 from files in a directory.

This allows you to run queries on external data from Autonomous Database.

### Syntax

```
DBMS_CLOUD.CREATE_EXTERNAL_TABLE (
    table_name       IN VARCHAR2,
    credential_name  IN VARCHAR2 DEFAULT NULL,
    file_uri_list    IN CLOB,
    column_list      IN CLOB,
    field_list       IN CLOB DEFAULT,
    format           IN CLOB DEFAULT);
```

### Parameters

| Parameter | Description |
|---|---|
| `table_name` | The name of the external table. |
| `credential_name` | The name of the credential to access the Cloud Object Storage.<br>This parameter is not used when you specify a directory with `file_uri_list`. |
| `file_uri_list` | This parameter specifies either a comma-delimited list of source file URIs or one or more directories and source files.<br>**Cloud source file URIs**<br>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.<br>The format of the URIs depends on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats.<br>**Directory**<br>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.<br>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?'`<br>To specify multiple directories, use a comma separated list of directories: For example:`'MY_DIR1:*, MY_DIR2:test?'`<br>Use double quotes to specify a case-sensitive directory name. For example:`'"my_dir1":*, "my_dir2":Test?'`<br>To include a quote character, use two quotes. For example:`'MY_DIR:''filename.ext'`. This specifies the `filename` starts with a quote (`'`). |

| Parameter | Description |
|---|---|
| column_list | (Optional) This field, when specified, overrides the format->schema parameter which specifies that the schema, columns, and data types, are derived automatically. See the format parameter for details.<br>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.<br>For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details.<br>For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping for details.<br>For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details. |
| field_list | Ignored for Avro, ORC, or Parquet files.<br>The fields in the source match the external table columns by name. Source data types are converted to the external table column data type.<br>For ORC files, see DBMS_CLOUD Package ORC to Oracle Data Type Mapping<br>For Parquet files, see DBMS_CLOUD Package Parquet to Oracle Data Type Mapping for details.<br>For Avro files, see DBMS_CLOUD Package Avro to Oracle Data Type Mapping for details. |
| format | For Avro, ORC, or Parquet, there are only two supported parameters. 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 Mappingfor information on column name mapping and column name conversion usage in Oracle SQL.

## CREATE_HYBRID_PART_TABLE Procedure

This procedure creates a hybrid partitioned table. This allows you to run queries on hybrid partitioned data from Autonomous Database using database objects and files in the Cloud, or database objects and files in a directory.

### Syntax

```
DBMS_CLOUD.CREATE_HYBRID_PART_TABLE (
    table_name           IN VARCHAR2,
    credential_name      IN VARCHAR2,
    partitioning_clause  IN CLOB,
    column_list          IN CLOB,
    field_list           IN CLOB DEFAULT,
    format               IN CLOB DEFAULT);
```

### Parameters

| Parameter | Description |
|---|---|
| `table_name` | The name of the external table. |
| `credential_name` | The name of the credential to access the Cloud Object Storage. |
| `partitioning_clause` | Specifies the complete partitioning clause, including the location information for individual partitions.<br>To use directories, the partitioning clause supports the `LOCATION` and `DEFAULT DIRECTORY` values. |
| `column_list` | Comma-delimited list of column names and data types for the external table. |
| `field_list` | Identifies the fields in the source files and their data types. The default value is NULL meaning the fields and their data types are determined by the *column_list* parameter. This argument's syntax is the same as the `field_list` clause in regular Oracle external tables. For more information about `field_list` see *Oracle® Database Utilities*. |
| `format` | The options describing the format of the source files. For the list of the options and how to specify the values see Format Parameter. |

### Usage Notes

- The procedure `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE` supports external partitioned files in the supported cloud object storage services, including:
  - Oracle Cloud Infrastructure Object Storage
  - Azure Blob Storage
  - Amazon S3

  - Google Cloud 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', 'characterset' value 'us7ascii'),
      column_list => 'col1 number, col2 number, col3 number',
      partitioning_clause => 'partition by range (col1)
                                (partition p1 values less than (1000)
external location
                                    ( ''&base_URL/file_11.txt'')
                                ,
                                partition p2 values less than (2000)
external location
                                    ( ''&base_URL/file_21.txt'')
                                ,
                                partition p3 values less than (3000)
                                )'
     );
   END;
/


BEGIN
   DBMS_CLOUD.CREATE_HYBRID_PART_TABLE(
    table_name  => 'HPT1',
    format      => json_object('delimiter'value ',', 'recorddelimiter'value
'newline'),
    column_list => 'NAME VARCHAR2(30), GENDER VARCHAR2(10), BALANCE number',
    partitioning_clause => 'partition by range (B  2  ALANCE)
               (partition p1 values less than (1000) external DEFAULT
DIRECTORY DATA_PUMP_DIR LOCATION (''Scott_male_1000.csv''),
                partition p2 values less than (2000) external DEFAULT
DIRECTORY DATA_PUMP_DIR LOCATION (''Mary_female_3000.csv''),
                partition p3 values less than (3000))' );
```

```
END;
/
```

## DELETE_ALL_OPERATIONS Procedure

This procedure clears either all data load operations logged in the `user_load_operations` table in your schema or clears all the data load operations of the specified type, as indicated with the `type` parameter.

### Syntax

```
DBMS_CLOUD.DELETE_ALL_OPERATIONS (
    type      IN VARCHAR DEFAULT NULL);
```

### Parameters

| Parameter | Description |
|---|---|
| `type` | Specifies the type of operation to delete. Type values can be found in the `TYPE` column in the `user_load_operations` table.<br>If no `type` is specified all rows are deleted. |

### Usage Note

- `DBMS_CLOUD.DELETE_ALL_OPERATIONS` does not delete currently running operations (operations in a "Running" status).

## DELETE_FILE Procedure

This procedure removes the specified file from the specified directory on Autonomous Database.

### Syntax

```
DBMS_CLOUD.DELETE_FILE (
      directory_name     IN VARCHAR2,
      file_name          IN VARCHAR2,
      force              IN BOOLEAN DEFAULT FALSE);
```

### Parameters

| Parameter | Description |
|---|---|
| `directory_name` | The name of the directory on the Autonomous Database instance. |
| `file_name` | The name of the file to be removed. |
| `force` | Ignore and do not report errors if the file does not exist. Valid values are: `TRUE` and `FALSE`. The default value is `FALSE`. |

> **Note:**
>
> To run `DBMS_CLOUD.DELETE_FILE` with a user other than ADMIN you need to grant write privileges on the directory that contains the file to that user. For example, run the following command as ADMIN to grant write privileges to `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;
/
```

# EXPORT_DATA Procedure

This procedure exports data from Autonomous Database based on the result of a query. This procedure is overloaded and supports writing files to the cloud or to a directory.

Based on the `format type` parameter, the procedure exports files to the Cloud or to a directory location as text files in CSV, JSON, or XML format.

### Syntax

```
DBMS_CLOUD.EXPORT_DATA (
      file_uri_list     IN CLOB,
      format            IN CLOB,
      credential_name   IN VARCHAR2 DEFAULT NULL,
      query             IN CLOB);

DBMS_CLOUD.EXPORT_DATA (
      file_uri_list     IN CLOB DEFAULT NULL,
      format            IN CLOB DEFAULT NULL,
      credential_name   IN VARCHAR2 DEFAULT NULL,
      query             IN CLOB DEFAULT NULL,
      operation_id      OUT NOCOPY NUMBER);
```

### Parameters

| Parameter | Description |
|---|---|
| `credential_name` | The name of the credential to access the Cloud Object Storage.<br>When the credential parameter is not included, this specifies output to a directory. |
| `file_uri_list` | There are different forms, depending on the value of the format parameter and depending on whether you include a credential parameter:<br>• When the `format` parameter `type` value is `json`: The JSON on Object Store or to the specified directory location is saved with a generated file name based on the value of the `file_uri_list` parameter. See File Naming for Text Output (CSV,JSON, or XML) for more information.<br>• When the `credential_name` parameter is not specified you provide a directory name in `file_uri_list`.<br>The format of the URIs depend on the Cloud Object Storage service you are using, for details see Cloud Object Storage URI Formats. |
| `format` | A JSON string that provides export format options.<br>Supported option is:<br>`type`: The `type format` option is required and must have one of the values: `csv` \| `json` \| `xml`.<br>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, or XML. For example:<br>SELECT warehouse_id, quantity FROM inventories<br>When the format type value is json, each query result is checked and if it is not JSON, as determined with the function: JSON_OBJECT_T.parse(), DBMS_CLOUD.EXPORT_DATA transforms the query to include JSON_OBJECT function to convert the row into JSON. See JSON_OBJECT_T Object Type for more information.<br>For example:<br>SELECT JSON_OBJECT(* RETURNING CLOB) from(SELECT warehouse_id, quantity FROM inventories) |
| operation_id | Use this parameter to track the progress and final status of the export operation as the corresponding ID in the USER_LOAD_OPERATIONS view. |

**Usage Notes:**

- The query parameter value that you supply can be an advanced query, if required, such as a query that includes joins or subqueries.
- Depending on the format parameter specified, DBMS_CLOUD.EXPORT_DATA outputs the results of the specified query on the Cloud Object Store or to a directory location in CSV, JSON, or XML 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, 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 Database instance.

  See File Naming for Text Output (CSV, JSON, 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.

**Usage Notes for DBMS_CLOUD.EXPORT_DATA with Output to a Directory**

- The provided directory must exist and you must be logged in as the ADMIN user or have WRITE access to the directory.
- DBMS_CLOUD.EXPORT_DATA does not create directories.
- The procedure does not overwrite files. For example, if a dump file in the file_uri_list exists, DBMS_CLOUD.EXPORT_DATA reports an error such as:

```
ORA-31641: unable to create dump file  "/u02/exports/123.dmp"
ORA-27038: created file already exists
```

### Examples

The following example shows `DBMS_CLOUD.EXPORT_DATA` with the format `type` parameter with the value `datapump`:

```
BEGIN
   DBMS_CLOUD.EXPORT_DATA(
      credential_name =>'OBJ_STORE_CRED',
      file_uri_list =>'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp1.dmp',
      format => json_object('type' value 'datapump', 'compression'
value 'basic', 'version' value 'latest'),
      query => 'SELECT warehouse_id, quantity FROM inventories'
     );
   END;
/
```

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

The following example shows `DBMS_CLOUD.EXPORT_DATA` with the format `type` parameter with the value `json`:

```
BEGIN
   DBMS_CLOUD.EXPORT_DATA(
      credential_name => 'OBJ_STORE_CRED',
      file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/
exp1.json',
      query           => 'SELECT * FROM DEPT',
      format          => JSON_OBJECT('type' value 'json',
'compression' value 'gzip'));
     );
   END;
/
```

The following example shows `DBMS_CLOUD.EXPORT_DATA` with the format `type` parameter with the value `xml`:

```
BEGIN
   DBMS_CLOUD.EXPORT_DATA(
      credential_name => 'OBJ_STORE_CRED',
      file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp1.xml',
      query           => 'SELECT * FROM DEPT',
      format          => JSON_OBJECT('type' value 'xml', 'compression'
value 'gzip'));
     );
   END;
/
```

The following example shows `DBMS_CLOUD.EXPORT_DATA` with the format `type` parameter with the value `csv`:

```
BEGIN
   DBMS_CLOUD.EXPORT_DATA(
      credential_name => 'OBJ_STORE_CRED',
      file_uri_list   => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o/exp.csv',
      query           => 'SELECT * FROM DEPT',
      format          => JSON_OBJECT('type' value 'csv', 'delimiter' value
'|', 'compression' value 'gzip', 'header' value true ));
     );
   END;
/
```

## GET_OBJECT Procedure and Function

This procedure is overloaded. The procedure form reads an object from Cloud Object Storage and copies it to Autonomous Database. The function form reads an object from Cloud Object Storage and returns a `BLOB` to Autonomous Database.

### Syntax

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


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

### Parameters

| Parameter | Description |
|---|---|
| `credential_name` | The name of the credential to access the Cloud Object Storage. |
| `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.<br>1 |

| Parameter | Description |
|---|---|
| file_name | Specifies the name of the file to create. If file name is not specified, the file name is taken from after the last slash in the object_uri parameter. For special cases, for example when the file name contains slashes, use the file_name parameter. |
| startoffset | The offset, in bytes, from where the procedure starts reading. |
| endoffset | The offset, in bytes, until where the procedure stops reading. |
| compression | Specifies the compression used to store the object. When compression is set to 'AUTO' the file is uncompressed (the value 'AUTO' implies the object specified with object_uri is compressed with Gzip). |

**Note:**

To run DBMS_CLOUD.GET_OBJECT with a user other than ADMIN you need to grant WRITE privileges on the directory to that user. For example, run the following command as ADMIN to grant write privileges to 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. |

### 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.

### Example

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

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

OBJECT_NAME       BYTES   CHECKSUM      CREATED             LAST_MODIFIED
------------ ---------- ----------    ---------------------
```

```
---------------------
cwallet.sso         2965                2018-12-12T18:10:47Z
2019-11-23T06:36:54Z
```

## LIST_OBJECTS Function

This function lists objects in the specified location on object store. The results include the object names and additional metadata about the objects such as size, checksum, creation timestamp, and the last modification timestamp.

### Syntax

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

### Parameters

| Parameter | Description |
|---|---|
| credential_name | The name of the credential to access the Cloud Object Storage. |
| 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 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 |
| 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.<br>If you do not supply a source_credential_name value, the credential_name is set to NULL. |

| Parameter | Description |
|---|---|
| source_object_uri | Specifies URI, that point to the source Object Storage bucket or folder location.<br>This parameter is mandatory.<br>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.<br>This parameter is mandatory.<br>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.<br>If you do not supply a target_credential_name value, the target_object_uri is set to the source_credential_name value. |

**Example**

```
BEGIN
DBMS_CLOUD.MOVE_OBJECT (
    source_credential_name => 'OCI_CRED',
    source_object_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname1/bgfile.csv',
    target_object_uri    => 'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname2/myfile.csv'
);
END;
/
```

## PUT_OBJECT Procedure

This procedure is overloaded. In one form the procedure copies a file from Autonomous Database to the Cloud Object Storage. In another form the procedure copies a BLOB from Autonomous Database to the Cloud Object Storage.

**Syntax**

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


DBMS_CLOUD.PUT_OBJECT (
       credential_name      IN VARCHAR2,
       object_uri           IN VARCHAR2,
       contents             IN BLOB,
       file_name            IN VARCHAR2);
```

### Parameters

| Parameter | Description |
|---|---|
| credential_name | The name of the credential to access the Cloud Object Storage. |
| 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 Database. [1] |
| file_name | The name of the file in the specified directory. |

> **Note:**
>
> To run DBMS_CLOUD.PUT_OBJECT with a user other than ADMIN you need to grant read privileges on the directory to that user. For example, run the following command as ADMIN to grant read privileges to 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 |
|---|---|
| `table_name` | The name of the target table. The target table needs to be created before you run `DBMS_CLOUD.SYNC_EXTERNAL_PART_TABLE`. |
| `schema_name` | The name of the schema where the target table resides. The default value is NULL meaning the target table is in the same schema as the user running the procedure. |
| `update_columns` | The new files may introduce a change to the schema. Updates supported include: new columns, deleted columns. Updates to existing columns, for example a change in the data type throw errors.<br>Default Value: False |

## VALIDATE_EXTERNAL_PART_TABLE Procedure

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

### Syntax

```
DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
       table_name                IN VARCHAR2,
       partition_name            IN CLOB DEFAULT,
       subpartition_name         IN CLOB DEFAULT,
       schema_name               IN VARCHAR2 DEFAULT,
       rowcount                  IN NUMBER DEFAULT,
       partition_key_validation  IN BOOLEAN DEFAULT,
```

```
       stop_on_error              IN BOOLEAN DEFAULT);


DBMS_CLOUD.VALIDATE_EXTERNAL_PART_TABLE (
       table_name                 IN VARCHAR2,
       operation_id               OUT NUMBER,
       partition_name             IN CLOB DEFAULT,
       subpartition_name          IN CLOB DEFAULT,
       schema_name                IN VARCHAR2 DEFAULT,
       rowcount                   IN NUMBER DEFAULT,
       partition_key_validation   IN BOOLEAN DEFAULT,
       stop_on_error              IN BOOLEAN DEFAULT);
```

### Parameters

| Parameter | Description |
|---|---|
| table_name | The name of the external table. |
| operation_id | Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view. |
| partition_name | If defined, then only a specific partition is validated. If not specified then read all partitions sequentially until rowcount is reached. |
| subpartition_name | If defined, then only a specific subpartition is validated. If not specified then read from all external partitions or subpartitions sequentially until rowcount is reached. |
| schema_name | The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure. |
| rowcount | Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned. |
| partition_key_validation | For internal use only. Do not use this parameter. |
| stop_on_error | Determines if the validate should stop when a row is rejected. The default value is TRUE meaning the validate stops at the first rejected row. Setting the value to FALSE specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the rowcount parameter. |

## VALIDATE_EXTERNAL_TABLE Procedure

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

### Syntax

```
DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE (
    table_name      IN VARCHAR2,
    schema_name     IN VARCHAR2 DEFAULT,
    rowcount        IN NUMBER DEFAULT,
```

```
    stop_on_error   IN BOOLEAN DEFAULT);


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

### Parameters

| Parameter | Description |
|---|---|
| table_name | The name of the external table. |
| operation_id | Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view. |
| schema_name | The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure. |
| rowcount | Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned. |
| stop_on_error | Determines if the validate should stop when a row is rejected. The default value is TRUE meaning the validate stops at the first rejected row. Setting the value to FALSE specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the rowcount parameter.<br>If the external table refers to Avro, ORC, or Parquet files then the validate stops at the first rejected row.<br>When the external table specifies the format parameter type set to the value avro, orc, or parquet, the parameter stop_on_error effectively always has the value TRUE. Thus, the table badfile will always be empty for an external table referring to Avro, ORC, or Parquet files. |

### Usage Notes

- DBMS_CLOUD.VALIDATE_EXTERNAL_TABLE works with both partitioned external tables and hybrid partitioned tables. This potentially reads data from all external partitions until rowcount is reached or stop_on_error applies. You do not have control over which partition, or parts of a partition, is read in which order.

## VALIDATE_HYBRID_PART_TABLE Procedure

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

### Syntax

```
DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
       table_name                  IN VARCHAR2,
       partition_name              IN CLOB DEFAULT,
       subpartition_name           IN CLOB DEFAULT,
       schema_name                 IN VARCHAR2 DEFAULT,
       rowcount                    IN NUMBER DEFAULT,
       partition_key_validation    IN BOOLEAN DEFAULT,
       stop_on_error               IN BOOLEAN DEFAULT);


DBMS_CLOUD.VALIDATE_HYBRID_PART_TABLE (
       table_name                  IN VARCHAR2,
       operation_id                OUT NUMBER,
       partition_name              IN CLOB DEFAULT,
       subpartition_name           IN CLOB DEFAULT,
       schema_name                 IN VARCHAR2 DEFAULT,
       rowcount                    IN NUMBER DEFAULT,
       partition_key_validation    IN BOOLEAN DEFAULT,
       stop_on_error               IN BOOLEAN DEFAULT);
```

### Parameters

| Parameter | Description |
|---|---|
| table_name | The name of the external table. |
| operation_id | Use this parameter to track the progress and final status of the load operation as the corresponding ID in the USER_LOAD_OPERATIONS view. |
| partition_name | If defined, then only a specific partition is validated. If not specified then read from all external partitions sequentially until rowcount is reached. |
| subpartition_name | If defined, then only a specific subpartition is validated. If not specified then read from all external partitions or subpartitions sequentially until rowcount is reached. |
| schema_name | The name of the schema where the external table resides. The default value is NULL meaning the external table is in the same schema as the user running the procedure. |
| rowcount | Number of rows to be scanned. The default value is NULL meaning all the rows in the source files are scanned. |
| partition_key_validation | For internal use only. Do not use this parameter. |
| stop_on_error | Determines if the validate should stop when a row is rejected. The default value is TRUE meaning the validate stops at the first rejected row. Setting the value to FALSE specifies that the validate does not stop at the first rejected row and validates all rows up to the value specified for the rowcount parameter. |

# DBMS_CLOUD REST APIs

This section covers the `DBMS_CLOUD` REST APIs provided with Autonomous Database on Dedicated Exadata Infrastructure.

## Prerequisites

As a developer, you can use DBMS_CLOUD REST APIs with Autonomous Databases deployed on Oracle Public Cloud 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
- Exadata Cloud@Customer


### Oracle Public Cloud

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 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 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.

## DBMS_CLOUD REST APIs

This section covers the `DBMS_CLOUD` REST APIs provided with Autonomous Database.

| REST API | Description |
|---|---|
| GET_RESPONSE_HEADERS Function | This function returns the HTTP response headers as JSON data in a JSON object in Autonomous Database. |
| GET_RESPONSE_TEXT Function | This function returns the HTTP response in TEXT format (`VARCHAR2` or `CLOB`) in Autonomous Database. Usually, most Cloud REST APIs return JSON response in text format. This function is useful if you expect the HTTP response is in text format. |
| GET_API_RESULT_CACHE_SIZE Function | This function returns the configured result cache size. |
| SEND_REQUEST Function and Procedure | This function begins an HTTP request, gets the response, and ends the response in Autonomous Database. This function provides a workflow for sending a Cloud REST API request with arguments and a return response code and payload. |
| SET_API_RESULT_CACHE_SIZE Procedure | This procedure sets the maximum cache size for current session. |

## DBMS_CLOUD REST API Overview

When you use PL/SQL in your application and you need to call Cloud REST APIs you can use `DBMS_CLOUD.SEND_REQUEST` to send the REST API requests.

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 |
|---|---|---|
| METHOD_DELETE | VARCHAR2(6) | 'DELETE' |
| METHOD_GET | VARCHAR2(3) | 'GET' |
| METHOD_HEAD | VARCHAR2(4) | 'HEAD' |
| METHOD_POST | VARCHAR2(4) | 'POST' |
| METHOD_PUT | VARCHAR2(3) | 'PUT' |

## DBMS_CLOUD REST API Results Cache

You can save `DBMS_CLOUD` REST API results when you set the `cache` parameter to true with `DBMS_CLOUD.SEND_REQUEST`. The `SESSION_CLOUD_API_RESULTS` view describes the columns you can use when REST API results are saved.

By default `DBMS_CLOUD` REST API calls do not save results for your session. In this case you use the `DBMS_CLOUD.SEND_REQUEST` function to return results.

When you use `DBMS_CLOUD.SEND_REQUEST` and set the `cache` parameter to `TRUE`, results are saved and you can view past results in the `SESSION_CLOUD_API_RESULTS` view. Saving and querying historical results of `DBMS_CLOUD` REST API requests can help you when you need to work with your previous results in your applications.

[1] Support for Azure Cloud REST API calls is limited to the domain "blob.windows.net".

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 definers procedure's owner. For this case, by default when cache_scope is PRIVATE', the invoker's session cannot see the results.

  If the definer's procedure owner wants to make the results available to any invoking session user, then they must set cache_scope to 'PUBLIC' in the DBMS_CLOUD.SEND_REQUEST.

## DBMS_CLOUD REST API SESSION_CLOUD_API_RESULTS View

You can save DBMS_CLOUD REST API results when you set the cache parameter to true with DBMS_CLOUD.SEND_REQUEST. The SESSION_CLOUD_API_RESULTS view describes the columns you can use when REST API results are saved.

The view SESSION_CLOUD_API_RESULTS is the view created if you cache results with DBMS_CLOUD.SEND_REQUEST. You can query historical results which belong to your user session. When the session ends, the data in the SESSION_CLOUD_API_RESULTS is purged.

| Column | Description |
|---|---|
| URI | The DBMS_CLOUD REST API request URL |
| TIMESTAMP | The DBMS_CLOUD REST API response timestamp |
| CLOUD_TYPE | The DBMS_CLOUD REST API cloud type, such as Oracle Cloud Infrastructure and AZURE_BLOB |
| REQUEST_METHOD | The DBMS_CLOUD REST API request method, such as GET, PUT, HEAD |
| REQUEST_HEADERS | The DBMS_CLOUD REST API request headers |
| REQUEST_BODY_TEXT | The DBMS_CLOUD REST API request body in CLOB |
| RESPONSE_STATUS_CODE | The DBMS_CLOUD REST API response status code, such as 200(OK), 404(Not Found) |
| RESPONSE_HEADERS | The DBMS_CLOUD REST API response headers |
| RESPONSE_BODY_TEXT | The DBMS_CLOUD REST API response body in CLOB |
| SCOPE | The cache_scope set by DBMS_CLOUD.SEND_REQUEST. Valid values are PUBLIC or PRIVATE. |

## GET_RESPONSE_HEADERS Function

This function returns the HTTP response headers as JSON data in a JSON object.

### Syntax

```
DBMS_CLOUD.GET_RESPONSE_HEADERS(
       resp          IN DBMS_CLOUD_TYPES.resp)
   RETURN JSON_OBJECT_T;
```

### Parameters

| Parameter | Description |
|---|---|
| resp | HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST. |

### Exceptions

| Exception | Error | Description |
|---|---|---|
| invalid_response | ORA-20025 | Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_HEADERS. |

## GET_RESPONSE_TEXT Function

This function returns the HTTP response in TEXT format (VARCHAR2 or CLOB). Usually, most Cloud REST APIs return JSON response in text format. This function is useful if you expect the the HTTP response is in text format.

### Syntax

```
DBMS_CLOUD.GET_RESPONSE_TEXT(
       resp          IN DBMS_CLOUD_TYPES.resp)
   RETURN CLOB;
```

### Parameters

| Parameter | Description |
|---|---|
| resp | HTTP Response type returned from DBMS_CLOUD.SEND_REQUEST. |

### Exceptions

| Exception | Error | Description |
|---|---|---|
| invalid_response | ORA-20025 | Invalid response type object passed to DBMS_CLOUD.GET_RESPONSE_TEXT. |

## GET_API_RESULT_CACHE_SIZE Function

This function returns the configured result cache size. The cache size value only applies for the current session.

### Syntax

```
DBMS_CLOUD.GET_API_RESULT_CACHE_SIZE()
   RETURN NUMBER;
```

## SEND_REQUEST Function and Procedure

This function and procedure begins an HTTP request, gets the response, and ends the response. This function provides a workflow for sending a cloud REST API request with arguments and the function returns a response code and payload. If you use the procedure,

you can view results and response details from the saved results with the SESSION_CLOUD_API_RESULTS view.

### Syntax

```
DBMS_CLOUD.SEND_REQUEST(
       credential_name    IN VARCHAR2,
       uri                IN VARCHAR2,
       method             IN VARCHAR2,
       headers            IN CLOB DEFAULT NULL,
       async_request_url  IN VARCHAR2 DEFAULT NULL,
       wait_for_states    IN DBMS_CLOUD_TYPES.wait_for_states_t
DEFAULT NULL,
       timeout            IN NUMBER DEFAULT 0,
       cache              IN PL/SQL BOOLEAN DEFAULT FALSE,
       cache_scope        IN VARCHAR2 DEFAULT 'PRIVATE',
       body               IN BLOB DEFAULT NULL)
   RETURN DBMS_CLOUD_TYPES.resp;

DBMS_CLOUD.SEND_REQUEST(
       credential_name    IN VARCHAR2,
       uri                IN VARCHAR2,
       method             IN VARCHAR2,
       headers            IN CLOB DEFAULT NULL,
       async_request_url  IN VARCHAR2 DEFAULT NULL,
       wait_for_states    IN DBMS_CLOUD_TYPES.wait_for_states_t
DEFAULT NULL,
       timeout            IN NUMBER DEFAULT 0,
       cache              IN PL/SQL BOOLEAN DEFAULT FALSE,
       cache_scope        IN VARCHAR2 DEFAULT 'PRIVATE',
       body               IN BLOB DEFAULT NULL);
```

### Parameters

| Parameter | Description |
|---|---|
| credential_name | The name of the credential for authenticating with the corresponding cloud native API. |
| uri | HTTP URI to make the request. |
| method | HTTP Request Method: GET, PUT, POST, HEAD, DELETE. Use the DBMS_CLOUD package constant to specify the method.<br>See DBMS_CLOUD REST API Constants for more information. |
| headers | HTTP Request headers for the corresponding cloud native API in JSON format. The authentication headers are set automatically, only pass custom headers. |
| async_request_url | An asynchronous request URL.<br>To obtain the URL select your request API from the list of APIs (see https://docs.cloud.oracle.com/en-us/iaas/api/). Then, navigate to find the API for your request in the left pane. For example, Database Services API → Autonomous Database → StopAutonomousDatabase. This page shows the API home (and shows the base endpoint). Then, append the base endpoint with the relative path obtained for your work request WorkRequest link. |

| 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'.<br>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.<br>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.<br>The default value is FALSE, which means REST API requests are not cached. |
| cache_scope | Specifies whether everyone can have access to this request result cache. Valid values: "PRIVATE" and "PUBLIC". The default value is "PRIVATE". |
| body | HTTP Request Body for PUT and POST requests. |

### Exceptions

| Exception | Error | Description |
|---|---|---|
| invalid_req_method | ORA-20023 | Request method passed to DBMS_CLOUD.SEND_REQUEST is invalid. |
| invalid_req_header | ORA-20024 | Request headers passed to DBMS_CLOUD.SEND_REQUEST are not in valid JSON format. |

### Usage Notes

- If you are using Oracle Cloud Infrastructure, you must use a Signing Key based credential value for the credential_name. See CREATE_CREDENTIAL Procedure for more information.
- The optional parameters async_request_url, wait_for_states, and timeout allow you to handle long running requests. Using this asynchronous form of send_request, the function waits for the completion status specified in wait_for_states before returning. With these parameters in the send request, you pass the expected return states in the wait_for_states parameter, and you use the async_request_url parameter to specify an associated work request, the request does not return immediately. Instead, the request probes the async_request_url until the return state is one of the expected states or the timeout is exceeded (timeout is optional). If no timeout is specified, the request waits until a state found in wait_for_states occurs.

## SET_API_RESULT_CACHE_SIZE Procedure

This procedure sets the maximum cache size for current session. The cache size value only applies for the current session.

### Syntax

```
DBMS_CLOUD.SET_API_RESULT_CACHE_SIZE(
       cache_size          IN NUMBER);
```

### Parameters

| Parameter | Description |
|---|---|
| `cache_size` | Set the maximum cache size to the specified value `cache_size`. If the new maximum cache size is smaller than the current cache size, older records are dropped until the number of rows is equal to the specified maximum cache size. The maximum value is `10000`.<br>If the cache size is set to `0`, caching is disabled in the session.<br>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);
```

# Format Parameter

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

The two ways to specify the format parameter are:

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

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 | Default |
|---|---|---|---|
| `blankasnull` | When set to `true`, loads fields consisting of spaces as null. | `blankasnull : true` | `False` |
| `characterset` | Specifies the characterset of source files | `characterset`: *string* | Database characterset |

| Format Option | Description | Syntax | Default |
|---|---|---|---|
| `compression` | Specifies the compression type of the source file.<br><br>**Note:** ZIP archiving format is not supported. | `compression: auto\|gzip\|zlib\|bzip2` | Specifying the value `auto` checks for the compression types: `gzip, zlib, bzip2`.<br><br>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` | `reject_record` |

| Format Option | Description | Syntax | Default |
|---|---|---|---|
| `dateformat` | Specifies the date format in the source file. The format option `AUTO` searches for the following formats:<br>`J`<br>`MM-DD-YYYYBC`<br>`MM-DD-YYYY`<br>`YYYYMMDD HHMISS`<br>`YYMMDD HHMISS`<br>`YYYY.DDD`<br>`YYYY-MM-DD` | `dateformat` : *string* | Database date format |
| `delimiter` | Specifies the field delimiter | `delimiter` : *character* | \| (pipe character) |
| `escape` | The character "\" is used as the escape character when specified. | `escape : true` | `False` |
| `ignoreblanklines` | Blank lines are ignored when set to true. | `ignoreblanklines : true` | `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` | `False` |
| `language` | Specifies a language name (for example, FRENCH), from which locale-sensitive information can be derived. | `language`: *string*<br>See Locale Data in *Oracle Database Globalization Support Guide* for a listing of Oracle-supported languages. | Null |
| `numericcharacters` | Specifies the characters to use as the group separator and decimal character.<br>*decimal_character*: The decimal separates the integer portion of a number from the decimal portion.<br>*group_separator*: The group separator separates integer groups (that is, thousands, millions, billions, and so on). | `numericcharacters: 'decimal_character group_separator'`<br>See NLS_NUMERIC_CHARACTERS in *Oracle Database Globalization Support Guide* for more information. | `"."` |
| `numberformat` | Specifies the number format model. Number format models cause the number to be rounded to the specified number of significant digits. A number format model is composed of one or more number format elements.<br>This is used in combination with `numericcharacters`. | `numberformat: number_format_model`<br>See Number Format Models in *SQL Language Reference* for more information. | Default is derived from the setting of the `NLS_TERRITORY` parameter |

| Format Option | Description | Syntax | Default |
|---|---|---|---|
| `quote` | Specifies the quote character for the fields, the quote characters are removed during loading when specified. | `quote`: *character* | Null meaning no quote |
| `recorddelimiter` | Specifies the record delimiter. | `recorddelimiter`: *character* | newline<br>By default, `DBMS_CLOUD` tries to automatically find the correct newline character as the delimiter. It first searches the file for the Windows newline character "`\r\n`". If it finds the Windows newline character, this is used as the record delimiter for the file. If a Windows newline character is not found, it searches for the UNIX/ Linux newline character "`\n`" and if it finds one it uses it as the record delimiter for the file.<br>Specify this parameter explicitly if you want to override the default behavior, for example:<br>`format => json_object('record delimiter' VALUE '''\r\n''')` |
| `rejectlimit` | The operation will error out after specified number of rows are rejected. | `rejectlimit`: *number* | 0 |
| `removequotes` | Removes any quotes that are around any field in the source file. | `removequotes`: true | `False` |
| `skipheaders` | Specifies how many rows should be skipped from the start of the file. | `skipheaders`: *number* | 0 if not specified, 1 if specified without a value |
| `territory` | Specifies a territory name to further determine input data characteristics. | `territory`: *string*<br>See Locale Data in *Oracle Database Globalization Support Guide* for a listing of Oracle-supported territories. | Null |

| Format Option | Description | Syntax | Default |
|---|---|---|---|
| `timestampformat` | Specifies the timestamp format in the source file. The format option `AUTO` searches for the following formats:<br>`YYYY-MM-DD HH:MI:SS.FF`<br>`YYYY-MM-DD`<br>`HH:MI:SS.FF3`<br>`MM/DD/YYYY HH:MI:SS.FF3` | `timestampformat` : *string* | Database timestamp format |
| `timestampltzformat` | Specifies the timestamp with local timezone format in the source file. The format option `AUTO` searches for the following formats:<br>`DD Mon YYYY HH:MI:SS.FF TZR`<br>`MM/DD/YYYY HH:MI:SS.FF TZR`<br>`YYYY-MM-DD HH:MI:SS+/-TZR`<br>`YYYY-MM-DD HH:MI:SS.FF3`<br>`DD.MM.YYYY HH:MI:SS TZR` | `timestampltzformat` : *string* | Database timestamp with local timezone format |
| `timestamptzformat` | Specifies the timestamp with timezone format in the source file. The format option `AUTO` searches for the following formats:<br>`DD Mon YYYY HH:MI:SS.FF TZR`<br>`MM/DD/YYYY HH:MI:SS.FF TZR`<br>`YYYY-MM-DD HH:MI:SS+/-TZR`<br>`YYYY-MM-DD HH:MI:SS.FF3`<br>`DD.MM.YYYY HH:MI:SS TZR` | `timestamptzformat`: *string* | Database timestamp with timezone format |
| `trimspaces` | Specifies how the leading and trailing spaces of the fields are trimmed.<br>See the description of `trim_spec` in Oracle Database Utilities. | `trimspaces`: `rtrim\| ltrim\| notrim\| lrtrim\| ldrtrim` | `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` | `False` |

| Format Option | Description | Syntax | Default |
|---|---|---|---|
| `type` | Specifies the source file type.<br>Cannot be specified together with `delimiter` or `quote`.<br>See the description of `CSV` in Oracle Database Utilities. | `type: csv\|csv with embedded\|csv without embedded`<br>`csv` is the same as `csv without embedded`. | Null |

# 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`, 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, 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.<br>Note: ZIP archiving format is not supported.<br>When the `format type` is `csv`, `json`, or `xml`, the default compression is Null, meaning no compression. | When the `type` is:`csv` \| `json` \| `xml`<br>`compression:gzip`<br>Default value: Null value meaning no compression. |

| Format Option | Description | Syntax |
|---|---|---|
| `delimiter` | Specifies a custom field delimiter.<br>`format => json_object('delimiter' value '\|')`<br>The delimiter value cannot be an ASCII code or an escape character.<br>**Note:** This option only applies with `csv` type. | `delimiter` : *character*<br>Default value , (comma) |
| `endquote` | Specifies that fields can be enclosed between two delimiters, with `quote` and `endquote`. If `endquote` is not specified, then the `quote` character will be used by default as the `endquote` character.<br>For example:<br>`format => JSON_OBJECT('quote' value '(', 'endquote' value ')')`<br>**Note:** This option only applies with `csv` type. | `endquote`:*character*<br>Default value: Null, meaning no `endquote`. |
| `escape` | Specifies the occurrence of quote character in the field value using `"\"` character.<br>**Note:** This option only applies with `csv` type. | `escape : true`<br>Default value: `false` |

| 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.<br>Use encryption to specify the following parameters to encrypt and decrypt:<br>• user_defined_function: Specifies a fully qualified user-defined function to decrypt or encrypt the specified BLOB (binary large object). It returns a decrypted or encrypted BLOB. This parameter is mutually exclusive with other parameters for encryption.<br>For example, ADMIN.DECRYPTION_CALLBACK.<br>• type: Specifies the built-in encryption algorithm to decrypt or encrypt. user_defined_function and type are mutually exclusive.<br>type accepts values in the *Block Cipher Algorithms* + *Block Cipher Chaining Modifiers* + *Block Cipher Padding Modifiers* format.<br>Supported Block Cipher Algorithms are:<br>– DBMS_CRYPTO.ENCRYPT_AES256<br>Supported Block Cipher Chaining Modifiers are:<br>– DBMS_CRYPTO.CHAIN_CBC<br>– DBMS_CRYPTO.CHAIN_CFB<br>– DBMS_CRYPTO.CHAIN_ECB<br>– DBMS_CRYPTO.CHAIN_OFB<br>Supported Block Cipher Padding Modifiers are:<br>– DBMS_CRYPTO.PAD_PKCS5<br>– DBMS_CRYPTO.PAD_NONE<br>– DBMS_CRYPTO.PAD_ZERO<br>– DBMS_CRYPTO.PAD_ORCL<br>• credential_name: Specifies the credential used to store the encryption key.<br>The Block Cipher Chaining Modifiers and Block Cipher Padding Modifiers values defaults to DBMS_CRYPTO.CHAIN_CBC and DBMS_CRYPTO.PAD_PKCS5, if you do not specify values for these parameters.<br>The format option encryption is used with the following DBMS_CLOUD procedures:<br>• Used to pass parameters to decrypt for these procedures:<br>– DBMS_CLOUD.COPY_DATA<br>– DBMS_CLOUD.CREATE_EXTERNAL_TABLE<br>– DBMS_CLOUD.CREATE_HYBRID_TABLE<br>– DBMS_CLOUD.COPY_COLLECTION<br>• Used to pass parameters to encrypt for these procedure: | |

| Format Option | Description | Syntax |
|---|---|---|
| | – DBMS_CLOUD.EXPORT_DATA<br>For example:<br>`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'))` | |
| `header` | Writes column names as the first line in output files of `csv` type.<br>The `header` option can accept a `boolean` or a `string` value.<br>The valid values are:<br>• `false`: Skips the header row.<br>• `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.<br>• *`String to define custom header names`*: Enables you to define header rows with custom names. The number of columns and delimiters in the string value must match the number of columns and delimiters in the `SELECT` statement. The default delimiter is comma (,).<br>For example:<br>`format => JSON_OBJECT('type' value 'csv', 'delimiter' value '|', 'compression' value 'gzip', 'header' value true)`<br>**Note:**<br>This option only applies with `csv type`. | `header: true\| false\| String to define custom header names`<br>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, or XML.<br>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.<br>If no file extension is desired, use the value: `fileextension ='none'` | Valid values: Any file extension.<br>Default value: Depends on the `format type` option:<br>• CSV format: `.csv`<br>• JSON format: `.json`<br>• XML format: `.xml` |

| Format Option | Description | Syntax |
|---|---|---|
| `maxfilesize` | Number in bytes for maximum size of output generated.<br>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`. | Minimum value: `10485760` (10 MB )<br>Maximum value: 1 GB<br>Default value: `10485760` (10 MB) |
| `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.<br>**Note:**<br>This option only applies with `csv type`. | `quote`: *character*<br>Default value: Null meaning do not enclose fields with quotes. |
| `trimspaces` | Specifies how the leading and trailing spaces of the fields are trimmed for CSV format. Trim spaces is applied before quoting the field, if the `quote` parameter is specified.<br>See the description of trim_spec.<br>**Note:**<br>This option only applies with `csv type`. | `trimspaces`: `rtrim\| ltrim\| notrim\| lrtrim\| ldrtrim`<br>Default value: `notrim` |
| `type` | Specifies the output file type.<br>`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.<br>`json`: specifies to export the query results as JSON files.<br>`xml`: specifies to export query results as rows of valid XML documents. Each row is encapsulated in a root XML tag of `<RECORD> </RECORD>`.<br>The query result is automatically transformed into XML format using XMLFOREST SQL function. Use Column Aliases to customize the XML tag names for columns.<br>See access_parameters Clause for more information. | `type`: `csv\|datapump\|json\|xml` |

# DBMS_CLOUD Avro, ORC, and Parquet Support

This section covers the `DBMS_CLOUD` Avro, ORC, and Parquet support provided with Autonomous Database.

## DBMS_CLOUD Package Format Options for Avro, ORC, or Parquet

The format argument in `DBMS_CLOUD` specifies the format of source files.

The two ways to specify the format argument are:

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

And:

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

Examples:

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

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

For example:

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

| Format Option | Description | Syntax |
|---|---|---|
| `type` | Specifies the file type. | `type : avro \| orc \| parquet` |

| Format Option | Description | Syntax |
|---|---|---|
| schema | When schema is set to first or all, the external table columns and data types are automatically derived from the Avro, ORC, or Parquet file metadata.<br>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.<br>The value first specifies to use the metadata from the first Avro, ORC, or Parquet file in the file_uri_list to auto generate the columns and their data types. Use first if all of the files have the same schema.<br>The value all specifies to use the metadata from all Avro, ORC, or Parquet files in the file_uri_list to auto generate the columns and their data types. Use all (slower) if the files may have different schemas.<br>**Default**: If column_list is specified, then the schema value, if specified is ignored. If column_list is not specified then the schema default value is first.<br>**Note**: For Avro, ORC, or Parquet format files the schema format option is not available and the column_list parameter must be specified for partitioned external tables using the DBMS_CLOUD.CREATE_EXTERNAL_PART_TABLE procedure. | schema : first \| all |

## DBMS_CLOUD Package Avro to Oracle Data Type Mapping

Describes the mapping of Avro data types to Oracle data types.

> **Note:**
>
> Complex types, such as maps, arrays, and structs are supported starting with Oracle Database 19c. See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Avro complex types.

| Avro Type | Oracle Type |
|---|---|
| INT | NUMBER(10) |
| LONG | NUMBER(19) |
| BOOL | NUMBER(1) |
| UTF8 BYTE_ARRAY | RAW(2000) |
| FLT | BINARY_FLOAT |
| DBL | BINARY_DOUBLE |
| DECIMAL(p) | NUMBER(p) |
| DECIMAL(p,s) | NUMBER(p,s) |
| DATE | DATE |
| STRING | VARCHAR2 |
| TIME_MILLIS | VARCHAR2(20 BYTE) |

| Avro Type | Oracle Type |
|---|---|
| TIME_MICROS | VARCHAR2(20 BYTE) |
| TIMESTAMP_MILLIS | TIMESTAMP(3) |
| TIMESTAMP_MICROS | TIMESTAMP(6) |
| ENUM | VARCHAR2(*n*) Where: "*n*" is the actual maximum length of the AVRO ENUM's possible values |
| DURATION | RAW(2000) |
| FIXED | RAW(2000) |
| NULL | VARCHAR2(1) BYTE |

See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Avro complex types.

## DBMS_CLOUD Package ORC to Oracle Data Type Mapping

Describes the mapping of ORC data types to Oracle data types.

See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using ORC complex types.

| ORC Type | Oracle Type | More Information |
|---|---|---|
| array | VARCHAR2(*n*) JSON format | DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types |
| bigint (64 bit) | NUMBER(19) | |
| binary | BLOB | |
| boolean (1 bit) | NUMBER(1) | |
| char | CHAR*(n)* | |
| date | DATE | |
| double | BINARY_DOUBLE | |
| float | BINARY_FLOAT | |
| int (32 bit) | NUMBER(10) | |
| list | VARCHAR2(*n*) JSON format | DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types |
| map | VARCHAR2(*n*) JSON format | DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types |
| smallint (16 bit) | NUMBER(5) | |
| string | VARCHAR2(4000) | |
| struct | VARCHAR2(*n*) JSON format | DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types |
| timestamp | TIMESTAMP | |
| tinyint (8 bit) | NUMBER(3) | |
| union | VARCHAR2(*n*) JSON format | DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types |
| varchar | VARCHAR2(*n*) | |

## DBMS_CLOUD Package Parquet to Oracle Data Type Mapping

Describes the mapping of Parquet data types to Oracle data types.

> **Note:**
>
> Complex types, such as maps, arrays, and structs are supported starting with Oracle Database 19c. See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Parquet complex types.

| Parquet Type | Oracle Type |
|---|---|
| UINT_64 | NUMBER(20) |
| INT_64 | NUMBER(19) |
| UINT_32 | NUMBER(10) |
| INT_32 | NUMBER(10) |
| UINT_16 | NUMBER(5) |
| INT_16 | NUMBER(5) |
| UINT_8 | NUMBER(3) |
| INT_8 | NUMBER(3) |
| BOOL | NUMBER(1) |
| UTF8 BYTE_ARRAY | VARCHAR2(4000 BYTE) |
| FLT | BINARY_FLOAT |
| DBL | BINARY_DOUBLE |
| DECIMAL(p) | NUMBER(p) |
| DECIMAL(p,s) | NUMBER(p,s) |
| DATE | DATE |
| STRING | VARCHAR2(4000) |
| TIME_MILLIS | VARCHAR2(20 BYTE) |
| TIME_MILLIS_UTC | VARCHAR2(20 BYTE) |
| TIME_MICROS | VARCHAR2(20 BYTE) |
| TIME_MICROS_UTC | VARCHAR2(20 BYTE) |
| TIMESTAMP_MILLIS | TIMESTAMP(3) |
| TIMESTAMP_MILLIS_UTC | TIMESTAMP(3) |
| TIMESTAMP_MICROS | TIMESTAMP(6) |
| TIMESTAMP_MICROS_UTC | TIMESTAMP(6) |
| TIMESTAMP_NANOS | TIMESTAMP(9) |

See DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types for information on using Parquet complex types.

## DBMS_CLOUD Package Oracle Data Type to Parquet Mapping

Describes the mapping of Oracle data types to Parquet data types.

| Oracle Type | Parquet Type |
|---|---|
| BINARY_DOUBLE | DBL |
| BINARY_FLOAT | FLT |
| DATE | DATE |
| NUMBER(p,s) | DECIMAL(p,s) |
| NUMBER(p) | DECIMAL(p) |
| TIMESTAMP(3) | TIMESTAMP_MILLIS |
| TIMESTAMP(3) | TIMESTAMP_MILLIS_UTC |
| TIMESTAMP(6) | TIMESTAMP_MICROS |
| TIMESTAMP(6) | TIMESTAMP_MICROS_UTC |
| TIMESTAMP(9) | TIMESTAMP_NANOS |
| VARCHAR2(4000) | STRING |

### NLS Session Parameters

The NLS session parameters `NLS_DATE_FORMAT`, `NLS_TIMESTAMP_FORMAT`, `NLS_TIMESTAMP_TZ_FORMAT` and `NLS_NUMERIC_CHARACTERS` define how the date, timestamp, timestamp with time zone format, and radix separator for timestamp with decimal marker should be shown when a table with those column types are queried.

In addition, when you export data using `DBMS_CLOUD.EXPORT_DATA` and specify Parquet output, Autonomous Database reads the values of these parameters from the `NLS_SESSION_PARAMETERS` table. Autonomous Database uses these values to convert the Oracle data types `DATE` or `TIMESTAMP` to Parquet types.

The `NLS_SESSION_PARAMETERS` parameters support an `RR` format mask (two character year specification).

The `RR` format mask for the year is not supported for these parameters when you export data to Parquet with `DBMS_CLOUD.EXPORT_DATA`. An application error is raised if you attempt to export to parquet and the `NLS_SESSION_PARAMETERS` are set to use the `RR` format mask (the default value for the `RR` format depends on the value of the `NLS_TERRITORY` parameter).

When one of the parameters `NLS_DATE_FORMAT`, `NLS_TIMESTAMP_FORMAT` or `NLS_TIMESTAMP_TZ_FORMAT` uses the `RR` format mask, you must change the format value to supported value to export data to Parquet with `DBMS_CLOUD.EXPORT_DATA`. For example:

```
ALTER SESSION SET NLS_DATE_FORMAT = "MM/DD/YYYY";
ALTER SESSION SET NLS_TIMESTAMP_FORMAT = 'YYYY-MM-DD HH:MI:SS.FF';
ALTER SESSION SET  NLS_TIMESTAMP_TZ_FORMAT='YYYY-MM-DD HH:MI:SS.FF TZH:TZM';
```

After you change the value, you can verify the change by querying the `NLS_SESSION_PARAMETERS` view:

```
SELECT value FROM NLS_SESSION_PARAMETERS
       WHERE parameter IN
('NLS_DATE_FORMAT','NLS_TIMESTAMP_FORMAT','NLS_TIMESTAMP_TZ_FORMAT');
```

If `NLS_DATE_FORMAT` is set, it applies to the columns with `DATE` datatype. If `NLS_TIMESTAMP_FORMAT` is set, it applies to the columns with `TIMESTAMP` datattype. If

NLS_TIMESTAMP_TZ_FORMAT is set, it applies to the columns with TIMESTAMP WITH TIME ZONE datatype.

See Date and Time Parameters and NLS Data Dictionary Views for more information.

## DBMS_CLOUD Package Avro, ORC, and Parquet Complex Types

Describes the mapping of Avro, ORC, and Parquet complex data types to Oracle data types.

Autonomous Database supports complex data types, including the following complex types:

- struct
- list
- map
- union
- array

When you specify a source file type of Avro, ORC, or Parquet and the source file includes complex columns, Autonomous Database queries return JSON for the complex columns. This simplifies processing of query results; you can use Oracle's powerful JSON parsing features consistently across the file types and data types. The following table shows the format for the complex types in Autonomous Database:

> **Note:**
>
> The complex fields map to VARCHAR2 columns and VARCHAR2 size limits apply.

| Type | Parquet | ORC | Avro | Oracle |
|---|---|---|---|---|
| List: sequence of values | List | List | Array | VARCHAR2 (JSON format) |
| Map: list of objects with single key | Map | Map | Map | VARCHAR2 (JSON format) |
| Union: values of different type | Not Available | Union | Union | VARCHAR2 (JSON format) |
| Object: zero or more key-value pairs | Struct | Struct | Record | VARCHAR2 (JSON format) |

If your ORC, Parquet, or Avro source files contain complex types, then you can query the JSON output for these common complex types. For example, the following shows an ORC file, movie-info.orc, with a complex type (the same complex type handling applies for Parquet and Avro source files).

Consider the movie-info.orc file with the following schema:

```
id    int
original_title string
overview       string
```

```
poster_path     string
release_date    string
vote_count      int
runtime         int
popularity      double
genres          array<struct<id:int,name:string>
```

Notice that each movie is categorized by multiple `genres` using an array of `genres`. The `genres` array is an `array` of `structs` and each item has an `id` (`int`) and a `name` (`string`). The `genres` array is considered a complex type. You can create a table over this ORC file using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` as follows:

```
BEGIN
DBMS_CLOUD.CREATE_EXTERNAL_TABLE(
        table_name =>'movie_info',
        credential_name =>'OBJ_STORE_CRED',
        file_uri_list =>'https://objectstorage.us-
phoenix-1.oraclecloud.com/n/mytenancy/b/movies/o/movie-info.orc',
        format => '{"type":"orc", "schema": "first"}');
END;
/
```

When you create the external table the database automatically generates the columns based on the schema in the ORC file (if you are using Avro or Parquet, the same applies). For this example, the `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` creates a table in your database as follows:

```
CREATE TABLE "ADMIN"."MOVIE_INFO"
    ( "ID"
      NUMBER(10,0),
      "ORIGINAL_TITLE"  VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "OVERVIEW"        VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "POSTER_PATH"     VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "RELEASE_DATE"    VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP",
      "VOTE_COUNT"      NUMBER(10,0),
      "RUNTIME"         NUMBER(10,0),
      "POPULARITY"      BINARY_DOUBLE,
      "GENRES"          VARCHAR2(4000 BYTE) COLLATE "USING_NLS_COMP"
    )  DEFAULT COLLATION "USING_NLS_COMP"
    ORGANIZATION EXTERNAL
     ( TYPE      ORACLE_BIGDATA
       DEFAULT DIRECTORY "DATA_PUMP_DIR"
       ACCESS PARAMETERS
       ( com.oracle.bigdata.credential.name=OBJ_STORE_CRED
         com.oracle.bigdata.fileformat=ORC
   )
       LOCATION
        (
      'https://objectstorage.us-phoenix-1.oraclecloud.com/n/mytenancy/b/
movies/o/movie-info.orc'
           )
     )
   REJECT LIMIT UNLIMITED
```

```
   PARALLEL;
 )
```

Now you can query the movie data:

```
SELECT original_title, release_date, genres
     FROM movie_info
     WHERE release_date > '2000'
     ORDER BY original_title;
```

This produces the following output:

```
original_title              release_date   genres
(500) Days of Summer        2009           [{"id":3,"name":"Drama"},
{"id":6,"name":"Comedy"},{"id":17,"name":"Horror"},
{"id":19,"name":"Western"},{"id":18,"name":"War"},
{"id":15,"name":"Romance"}]
10,000 BC                   2008           [{"id":6,"name":"Comedy"}]
11:14                       2003           [{"id":9,"name":"Thriller"},
{"id":14,"name":"Family"}]
127 Hours                   2010           [{"id":6,"name":"Comedy"},
{"id":3,"name":"Drama"}]
13 Going on 30              2004           [{"id":6,"name":"Comedy"},
{"id":3,"name":"Drama"},{"id":18,"name":"War"},
{"id":15,"name":"Romance"}]
1408                        2007           [{"id":45,"name":"Sci-Fi"},
{"id":6,"name":"Comedy"},{"id":17,"name":"Horror"},
{"id":6,"name":"Comedy"},{"id":18,"name":"War"}]
```

Notice that the complex type `genres` is returned as a JSON array.

To make the JSON data more useful, you can transform the column using Oracle's JSON functions. For example, you can use the JSON "." notation as well as the more powerful transform functions such as `JSON_TABLE`.

See Simple Dot-Notation Access to JSON Data for information on "." notation.

See SQL/JSON Function JSON_TABLE for information on `JSON_TABLE`.

The following example shows a query on the table that takes each value of the array and turns the value into a row in the result set:

```
SELECT original_title, release_date, m.genre_name, genres
    FROM movie_info mi,
       JSON_TABLE(mi.genres, '$.name[*]'
        COLUMNS (genre_name VARCHAR2(25) PATH
      '$')
                  ) AS m
 WHERE rownum < 10;
```

The `JSON_TABLE` creates a row for each value of the array, think outer join, and the struct is parsed to extract the name of the genre. This produces the following output:

```
original_title                  release_date         genre_name
genres
(500) Days of Summer            2009                 Drama
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                 Comedy
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                 Horror
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                 Western
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                 War
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
(500) Days of Summer            2009                 Romance
[{"id":3,"name":"Drama"},{"id":6,"name":"Comedy"},

{"id":17,"name":"Horror"},{"id":19,"name":"Western"},

{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
10,000 BC                       2008                 Comedy
[{"id":6,"name":"Comedy"}]
11:14                           2003                 Family
[{"id":9,"name":"Thriller"},{"id":14,"name":"Family"}]
11:14                           2003                 Thriller
[{"id":9,"name":"Thriller"},{"id":14,"name":"Family"}]
127 Hours                       2010                 Comedy
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"}]
127 Hours                       2010                 Drama
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"}]
13 Going on 30                  2004                 Romance
[{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"},
```

```
{"id":18,"name":"War"},{"id":15,"name":"Romance"}]
13 Going on 30                    2004
Comedy             [{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"},

 {"id":18,"name":"War"},{"id":15,"name":"Romance"}]
13 Going on 30                    2004
War                [{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"},

 {"id":18,"name":"War"},{"id":15,"name":"Romance"}]
13 Going on 30                    2004
Drama              [{"id":6,"name":"Comedy"},{"id":3,"name":"Drama"},

 {"id":18,"name":"War"},{"id":15,"name":"Romance"}]
```

## DBMS_CLOUD Package Avro, ORC, and Parquet to Oracle Column Name Mapping

Describes rules for how Avro, ORC, and Parquet column names are converted to Oracle column names.

The following are supported for Avro, ORC, and Parquet column names, but may require use of double quotes for Oracle SQL references in external tables. Thus, for ease of use and to avoid having to use double quotes when referencing column names, if possible do not use the following in Avro, ORC, and Parquet column names:

- Embedded blanks
- Leading numbers
- Leading underscores
- Oracle SQL reserved words

The following table shows various types of Avro, ORC, and Parquet column names, and rules for using the column names in Oracle column names in external tables.

| Avro, ORC, or Parquet Name | CREATE TABLE Name | Oracle CATALOG | Valid SQL | Notes |
|---|---|---|---|---|
| part, Part, or PART | part, Part, PART | PART | `select part`<br>`select Part`<br>`select paRt`<br>`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 |

| Avro, ORC, or Parquet Name | CREATE TABLE Name | Oracle CATALOG | Valid SQL | Notes |
|---|---|---|---|---|
| __index_key__ | "__index_key__" | __index_key__ | `select "__index_key__"` | Double quotes are required when there is a leading underscore, which also preserves the character case |
| 6Way | "6Way" | 6Way | `select "6Way"` | Double quotes are required when there is a leading numeric digit, which also preserves the character case |
| create, Create, or CREATE, and so on. (any case variation) partition, Partition, PARTITION, and so on (for an Oracle Reserved word) | "CREATE" "PARTITION" | CREATE PARTITION | `select "CREATE"`<br>`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"`<br>`select "Rowid"`<br>`select "ROWid"`<br>`select "rowid"` | For ROWID, any mixed or lower-case variation of ROWID preserves the case and must always be double-quoted and use the original case variations. Due to the inherent conflict with Oracle ROWID for the table, if you specify upper-case ROWID, it is automatically stored as lower-case "rowid" and must always be double-quoted when referenced. |

**Notes:**

- In general a column name in an external table can be referenced without double quotes.
- Unless there is an embedded blank, a leading underscore ("_") or leading numeric digit ("0" through "9") in the column name, the original case of the column name is preserved, and it must always be referenced with double quotes and using the original case (upper, lower or mixed-case) of the Avro, ORC, or Parquet column name.
- After using `DBMS_CLOUD.CREATE_EXTERNAL_TABLE` to create an external table with the format specified as `avro`, `orc`, or `parquet`, use the `DESCRIBE` command in SQL*Plus to view the table's column names.
- When Oracle SQL Reserved Words are used in Avro, ORC, or Parquet column names, they must always be double-quoted when referenced anywhere in SQL. See Oracle SQL Reserved Words for more information.

# DBMS_CLOUD Exceptions

The following table describes exceptions for `DBMS_CLOUD`.

| Exception | Code | Description |
|---|---|---|
| `reject_limit` | 20003 | The reject limit of an external table was reached. |
| `credential_not_exist` | 20004 | A credential object does not exist. |
| `table_not_exist` | 20005 | A table does not exist. |
| `unsupported_obj_store` | 20006 | An unsupported object store URI was provided. |
| `iden_too_long` | 20008 | An identifier is too long. |
| `invalid_format` | 20009 | A format argument is not valid. |
| `missing_credential` | 20010 | Mandatory credential object information was not specified. |
| `invalid_object_uri` | 20011 | An invalid object URI was provided. |
| `invalid_partitioning_clause` | 20012 | An partitioning clause is missing or was not provided. |
| `unsupported_feature` | 20013 | An unsupported feature was used that is not existent in the current database version. |
| `part_not_exist` | 20014 | A partition or subpartition does not exist, or a table is not a partitioned external table or hybrid partitioned table. |
| `invalid_table_name` | 20016 | An invalid table name was used. |
| `invalid_schema_name` | 20017 | An invalid schema name was used. |
| `invalid_dir_name` | 20018 | An invalid directory name was used. |
| `invalid_file_name` | 20019 | An invalid file name was used. |
| `invalid_cred_attribute` | 20020 | Invalid credential attributes were specified. |
| `table_exist` | 20021 | A table already exists. |
| `credential_exist` | 20022 | A credential object already exists. |
| `invalid_req_method` | 20023 | A request method is either too long or invalid. |

| Exception | Code | Description |
|---|---|---|
| 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 Database supports only standard-tier storage buckets. It does not support archive-tier storage buckets.

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure native URIs, with the format:

```
https://objectstorage.region.oraclecloud.com/n/namespace-string/b/bucket/o/
filename
```

For example, the Native URI for the file channels.txt in the *bucketname* bucket in the Phoenix data center is:

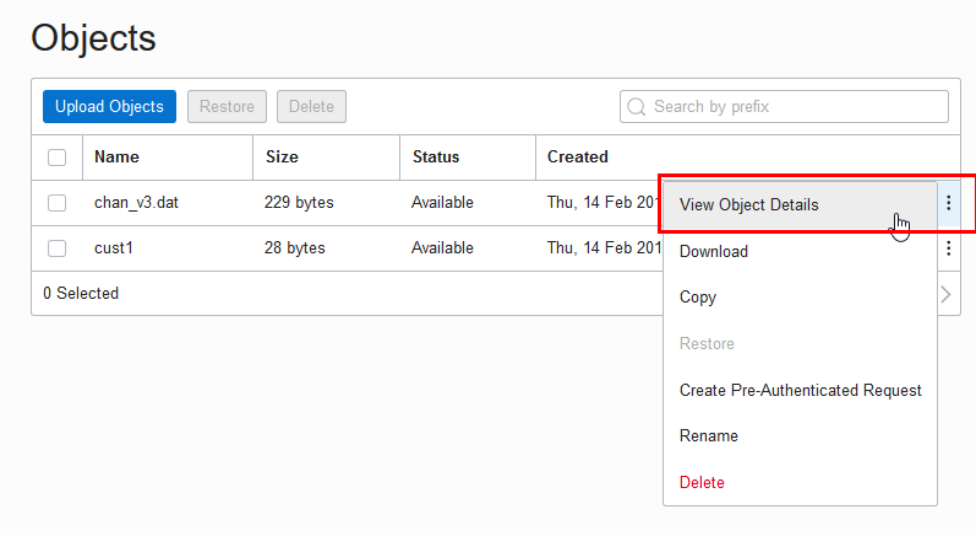
```
https://objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o/channels.txt
```

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

You can find the URI from the Oracle Cloud Infrastructure Object Storage "Object Details" in the right hand side ellipsis menu in the Object Store:

1. Open the Oracle Cloud Infrastructure Console by clicking the ☰ next to Oracle Cloud.
2. From the Oracle Cloud Infrastructure left navigation menu click **Core Infrastructure**. Under **Object Storage**, click **Object Storage**.
3. Under List Scope, select a **Compartment**.
4. From the **Name** column, select a bucket.

5. In the Objects area, click **View Object Details**.




6. On the **Object Details** page, the **URL Path (URI)** field shows the URI to access the object.

## Oracle Cloud Infrastructure Object Storage Swift URI Format

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure Swift URIs with the format:

`https://swiftobjectstorage.`*`region`*`.oraclecloud.com/v1/`*`namespace-string`*`/`*`bucket`*`/`*`filename`*

For example, the Swift URI for the file `channels.txt` in the *`bucketname`* bucket in the `Phoenix` data center is:

`https://swiftobjectstorage.us-phoenix-1.oraclecloud.com/v1/`*`namespace-string`*`/`*`bucketname`*`/channels.txt`

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

> **Note:**
>
> The source files need to be stored in an Object Storage tier bucket. Autonomous Database does not support buckets in the Archive Storage tier. See Overview of Object Storage for more information.

# Oracle Cloud Infrastructure Object Storage URI Format Using Pre-Authenticated Request URL

If your source files reside on the Oracle Cloud Infrastructure Object Storage you can use Oracle Cloud Infrastructure pre-authenticated URIs. When you create a pre-authenticated request, a unique URL is generated. You can then provide the unique URL to users in your organization, partners, or third parties to access the Object Storage resource target identified in the pre-authenticated request.

> **Note:**
>
> Carefully assess the business requirement for and the security ramifications of pre-authenticated access. When you create the pre-authenticated request URL, note the **Expiration** and the **Access Type** to make sure they are appropriate for your use.
> A pre-authenticated request URL gives anyone who has the URL access to the targets identified in the request for as long as the request is active. In addition to considering the operational needs of pre-authenticated access, it is equally important to manage its distribution.

The format for pre-authenticated request URLs is:

`https://objectstorage.`*`region`*`.oraclecloud.com/p/`*`encrypted_string`*`/n/`*`namespace-string`*`/b/`*`bucket`*`/o/`*`filename`*

For example, a sample pre-authenticated URI for the file `channels.txt` in the *bucketname* bucket in the `Phoenix` data center is:

`https://objectstorage.us-phoenix-1.oraclecloud.com/p/2xN-uDtWJNsiD910UCYGue/n/`*`namespace-string`*`/b/`*`bucketname`*`/o/channels.txt`

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

You can use a pre-authenticated URL in any `DBMS_CLOUD` procedure that takes a URL to access files in Oracle Cloud Infrastructure object store, without the need to create a credential. You need to either specify the `credential_name` parameter as `NULL` or not supply a `credential_name` parameter.

For example:

```
BEGIN
  DBMS_CLOUD.COPY_DATA(
     table_name =>'CHANNELS',
     file_uri_list =>'https://objectstorage.us-phoenix-1.oraclecloud.com/p/
unique-pre-authenticated-string/n/namespace-string/b/bucketname/o/
channels.txt',
     format => json_object('delimiter' value ',') );
END;
/
```

> **Note:**
>
> A list of mixed URLs is valid. If the URL list contains both pre-authenticated URLs and URLs that require authentication, `DBMS_CLOUD` uses the specified `credential_name` to access the URLs that require authentication and for the pre-authenticated URLs the specified `credential_name` is ignored.

See Using Pre-Authenticated Requests for more information.

## 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.

## Google Cloud Storage S3 Compatible URI Format

`DBMS_CLOUD` supports object storage service implementations for Google Cloud Storage with Amazon S3 compatible URLs.

> **Note:**
>
> To use `DBMS_CLOUD` with an Amazon S3 compatible object store you need to provide valid credentials. See #unique_532 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:

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.

# SODA Collection Metadata on Autonomous Database

Describes default and customized collection metadata on Autonomous Database.

## SODA Default Collection Metadata on Autonomous Database

Describes the default collection metadata on Autonomous Database, that is the metadata for a collection that is added when custom metadata is not supplied.

Each SODA implementation provides a way to create a default collection when you supply a collection name. For example, in SODA for Java you use the `createCollection` method and supply just a collection name parameter:

```
db.admin().createCollection("myCol");
```

This creates a collection with default collection metadata. When you create a default collection on your Autonomous Database, the collection metadata includes the following information (regardless of which SODA implementation you use to create the default collection):

```
{
   "keyColumn" :
   {
      "name" : "ID",
      "sqlType" : "VARCHAR2",
      "maxLength" : 255,
      "assignmentMethod" : "UUID"
   },

   "contentColumn" :
   {
      "name" : "JSON_DOCUMENT",
      "sqlType" : "BLOB",
      "jsonFormat" : "OSON"
   },
   "versionColumn" :
   {
     "name" : "VERSION",
     "method" : "UUID"
   },

   "lastModifiedColumn" :
   {
     "name" : "LAST_MODIFIED"
   },

   "creationTimeColumn" :
   {
      "name" : "CREATED_ON"
   },

   "readOnly" : false
}
```

## SODA Customized Collection Metadata on Autonomous Database

Describes SODA collection custom metadata on Autonomous Database.

Each SODA implementation provides a way to customize the collection metadata during collection creation. For example, in SODA for Java, you can use the following command:

```
OracleDocument metadata =
db.createDocumentFromString("metadata_string");
OracleCollection col = db.admin().createCollection("myCustomColl",
metadata);
```

In this example, for *metadata_string* you can use the default metadata as the starting point, and customize the following:

- Change `keyColumn.assignmentMethod` to `CLIENT`: Change the value of the `assignmentMethod` under `keyColumn` in the metadata to `CLIENT` (instead of `UUID`).

  Valid values for `keyColumn.assignmentMethod` on Autonomous Database:

  - **UUID** (default): Keys are generated by SODA, based on the `UUID`.
  - **CLIENT**: Keys are assigned by the client application.

- Provide a `mediaTypeColumn name` value: A media type column is needed if the collection is to be heterogeneous, that is, it can store documents other than JavaScript Object Notation (JSON). See Media Type Column Name for details.

The following example specifies client-assigned keys and a custom media type column. The `mediaTypeColumn name` is specified with the value *YOUR_MEDIA_TYPE_COLUMN_NAME*. Otherwise, the default settings are used.

```
{
   "keyColumn" :
   {
      "name" : "ID",
      "sqlType" : "VARCHAR2",
      "maxLength" : 255,
      "assignmentMethod" : "CLIENT"
   },

   "contentColumn" :
   {
      "name" : "JSON_DOCUMENT",
      "sqlType" : "BLOB"
   },

   "versionColumn" :
   {
     "name" : "VERSION",
     "method" : "UUID"
   },

   "lastModifiedColumn" :
   {
     "name" : "LAST_MODIFIED"
   },

   "creationTimeColumn" :
   {
      "name" : "CREATED_ON"
   },

   "mediaTypeColumn" :
   {
      "name" : "YOUR_MEDIA_TYPE_COLUMN_NAME"
   },

   "readOnly" : false
}
```

# Autonomous Database Views

## 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 |

# Summary of DBMS_CLOUD_ADMIN Subprograms

This section covers the DBMS_CLOUD_ADMIN subprograms provided with Autonomous Database.

| Subprogram | Description |
|---|---|
| ATTACH_FILE_SYSTEM Procedure | This procedure attaches a file system in a directory on your database. |

| Subprogram | Description |
|---|---|
| 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 Database instance. |
| ENABLE_EXTERNAL_AUTHENTICATION Procedure | This procedure enables a user to logon to Autonomous 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
);
```

### Parameters

| Parameter | Description |
|---|---|
| `file_system_name` | Specifies the name of the file system.<br>This parameter is mandatory. |
| `file_system_location` | Specifies the location of the file system.<br>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*.<br>For example:<br>• FQDN: `myhost.sub000445.myvcn.oraclevcn.com`<br>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.<br>• File Path: `/results`<br>This parameter is mandatory. |
| `directory_name` | Specifies the directory name for the attached file system. The directory must exist.<br>This parameter is mandatory. |
| `description` | Provides a description of the task.<br>This parameter is mandatory. |

**Example:**

```
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;
/
```

**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 NFS Version 3 to share
- If you attach to non-Oracle Cloud Infrastructure File Storage systems, the procedure supports NFS Version 2 and NFS Version 3
- `DBMS_CLOUD_ADMIN.ATTACH_FILE_SYSTEM` does not support NFS Version 4

# 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.<br>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.
```

## 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 .<br>• `'OCI_IAM'`<br>• `'AZURE_AD'`<br>• `'CMU'`<br>• `'KERBEROS'` |
| `force` | (Optional) Override a currently enabled external authentication scheme. Valid values are `TRUE` or `FALSE`.<br>The default value is `FALSE`. |

| Parameter | Description |
|---|---|
| params | A JSON string that provides additional parameters for the external authentication.<br>CMU parameters:<br>• location_uri: specifies the cloud storage URI for the bucket where files required for CMU are stored.<br>If you specify location_uri there is a fixed name directory object CMU_WALLET_DIR created in the database at the path 'cmu_wallet' to save the CMU configuration files. In this case, you do not need to supply the directory_name parameter.<br>• credential_name: specifies the credentials that are used to download the CMU configuration files from the Object Store to the directory object.<br>Default value is NULL which allows you to provide a Public, Preauthenticated, or pre-signed URL for Object Store bucket or subfolder.<br>• directory_name: specifies the directory name where configuration files required for CMU are stored. If directory_name is supplied, you are expected to copy the CMU configuration files dsi.ora and cwallet.sso to this directory object.<br>KERBEROS parameters:<br>• location_uri: specifies the cloud storage URI for the the bucket where files required for Kerberos are stored.<br>If you specify location_uri there is a fixed name directory object KERBEROS_DIR created in the database to save the Kerberos configuration files. In this case, you do not need to supply the directory_name parameter.<br>• credential_name: specifies the credential that are used to download Kerberos configuration files from the Object Store location to the directory object.<br>Default value is NULL which allows you to provide a Public, Preauthenticated, or pre-signed URL for Object Store bucket or subfolder.<br>• directory_name: specifies the directory name where files required for Kerberos are stored. If directory_name is supplied, you are expected to copy the Kerberos configuration files to this directory object.<br>AZURE_AD parameters:<br>• tenant_id: Tenant ID of the Azure Account. Tenant Id specifies the Autonomous Database instance's Azure AD application registration.<br>• application_id: Azure Application ID created in Azure AD to assign roles/schema mappings for external authentication in the Autonomous Database instance.<br>• application_id_uri: Unique URI assigned to the Azure Application.<br>This it the identifier for the Autonomous Database instance. The name must be domain qualified (this supports cross tenancy resource access).<br>The maximum length for this parameter is 256 characters. |

### Exceptions

| Exception | Error | Description |
|---|---|---|
| invalid_ext_auth | ORA-20004 | See the accompanying message for a detailed explanation. |

### Usage Notes

- With `type OCI_IAM`, if the resource principal is not enabled on the Autonomous Database instance, this routine enables resource principal with `DBMS_CLOUD_ADMIN.ENABLE_RESOURCE_PRINCIPAL`.
- This procedure sets the system parameters `IDENTITY_PROVIDER_TYPE` and `IDENTITY_PROVIDER_CONFIG` to required users to access the instance with Oracle Cloud Infrastructure Identity and Access Management authentication and authorization.

### Examples

#### Enable `OCI_IAM` Authentication

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'OCI_IAM',
     force=> TRUE );
END;
/

PL/SQL procedure successfully completed.
```

#### Enable `CMU` Authentication for Microsoft Active Directory

You pass in a directory name that contains the CMU configuration files through `params` JSON argument.

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'CMU',
     force => TRUE,
     params => JSON_OBJECT('directory_name' value 'CMU_DIR'); // CMU_DIR
directory object already exists
END;
/

PL/SQL procedure successfully completed.
```

You pass in a location URI pointing to an Object Storage location that contains CMU configuration files through `params` JSON argument.

```
BEGIN
   DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
       type     => 'CMU',
       params   => JSON_OBJECT('location_uri' value 'https://
objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/bucketname/o',
```

```
                                'credential_name' value
'my_credential_name')
   );
END;
/

PL/SQL procedure successfully completed.
```

### Enable Azure AD Authentication

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'AZURE_AD',
     force => TRUE,
     params   => JSON_OBJECT( 'tenant_id' VALUE '....',
                              'application_id' VALUE '...',
                              'application_id_uri' VALUE '.....' ));
END;
/

PL/SQL procedure successfully completed.
```

### Enable Kerberos Authentication

You pass in a directory name that contains Kerberos configuration files through `params` JSON argument.

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'KERBEROS',
     force => TRUE,
     params => JSON_OBJECT('directory_name' value 'KERBEROS_DIR'); //
KERBEROS_DIR directory object already exists
END;
/

PL/SQL procedure successfully completed.
```

You pass in a location URI pointing to an Object Storage location that contains Kerberos configuration files through `params` JSON argument.

```
BEGIN DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION(
     type => 'KERBEROS',
     force => TRUE,
     params => JSON_OBJECT('location_uri' value 'https://
objectstorage.us-phoenix-1.oraclecloud.com/n/namespace-string/b/
bucketname/o',
                           'credential_name' value
'my_credential_name');
END;
/

PL/SQL procedure successfully completed.
```

# DBMS_CLOUD_NOTIFICATION Package

The `DBMS_CLOUD_NOTIFICATION` package allows you to send messages or the output of a SQL query to a provider.

**Prerequisites**

- Oracle Public Cloud
- Exadata Cloud@Customer

## Oracle Public Cloud

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 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 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_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.<br>Valid values are: '`email`', '`msteams`', and '`slack`'<br>This parameter is mandatory. |
| `credential_name` | The name of the credential object to access the provider.<br>For the email provider, the credential is the name of the credential of the SMTP approved sender that contains its username and password.<br>For the msteams provider, the credential is the name of the credential.<br>For the Slack provider, the credential's username must be "`SLACK_TOKEN`" and the `password` is the Slack Bot Token.<br>This parameter is mandatory. |
| `query` | Specifies the SQL query to run. Results are sent to the provider.<br>This parameter is mandatory. |

| Parameter | Description |
|---|---|
| params | Specifies specific parameters for the provider in JSON format.<br>For the provider type email, the valid parameters are:<br>• sender. This specifies the Email ID of the approved sender in a String value.<br>• smtp_host. This specifies the SMTP host name in a String value.<br>• subject. This specifies the subject of the email in a String value The maximum size is 100 characters.<br>• recipient. This specifies the email IDs of recipients in a String value. Use a comma between email IDs when there are multiple recipients.<br>• to_cc. This specifies the email IDs that are receiving a CC of the email. It is a String value. Use a comma between email IDs when there are multiple CC recipients.<br>• to_bcc. This specifies the email IDs that are receiving a BCC of the email. It is a String value. Use a comma between email IDs when there are multiple BCC recipients.<br>• message. This specifies the message text in a String value.<br>• type. This specifies the output format in a String value as either CSV or JSON.<br>• title. This specifies the title of the attachment of SQL output in a String value. The title should only contain letters, digits, underscores, hyphens, or dots as characters in its value due to it being used to generate a file name.<br>For the provider type msteams, the valid parameters are:<br>• tenant. This specifies the tenant ID in a String format.<br>• team. This specifies the team ID in a String format.<br>• channel. This specifies the channel ID in a String format.<br>• title. This specifies the title of the file in a String format. The maximum size is 50 characters.<br>• type. This specifies the output format in a String value. Valid values are CSV or JSON.<br>For the provider type slack, the valid parameters are:<br>• channel. This specifies the Channel ID in a String value. The Channel ID is a unique ID for a channel and is different from the channel name. In Slack, when you view channel details you can find the Channel ID on the “About” tab.<br>• type. This specifies the output format in a String value. Valid values are CSV or JSON.<br>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.<br>Valid values are: 'email', 'msteams', and 'slack'<br>This parameter is mandatory. |
| credential_name | The name of the credential object to access the provider.<br>For the email provider, the credential is the name of the credential of the SMTP approved sender that contains its username and password.<br>For the msteams provider, the credential must contain the bot_id and the bot_secret key in both the username and the password.<br>For the Slack provider, the credential's username must be "SLACK_TOKEN" and the password is the Slack Bot Token.<br>This parameter is mandatory. |
| message | Specifies the message text.<br>This parameter is mandatory. |
| params | Specifies specific parameters for the provider in JSON format.<br>For the provider type email, the valid parameters are:<br>• sender. This specifies the Email ID of the approved sender in a String value.<br>• smtp_host. This specifies the SMTP host name in a String value.<br>• subject. This specifies the subject of the email in a String value. The maximum size is 100 characters.<br>• recipient. This specifies the email IDs of recipients in a String value. Use a comma between email IDs when there are multiple recipients.<br>• to_cc. This specifies the email IDs that are receiving a CC of the email. Use a comma between email IDs when there are multiple CC recipients.<br>• to_bcc. This specifies the email IDs that are receiving a BCC of the email. Use a comma between email IDs when there are multiple BCC recipients.<br>For the provider type msteams, the valid parameter is:<br>• channel. This specifies the Channel ID in a String value.<br>For the provider type slack, the valid parameter is:<br>• channel. This specifies the Channel ID in a String value.<br>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.<br>This parameter is mandatory. |

### Usage Notes

Use the procedure DBMS_CLOUD.CREATE_CREDENTIAL(credential_name, username, password) to create the credential object. See #unique_532 for more information.

- For the email provider, the user requires an SMTP connection endpoint for the Email Delivery server to obtain smtp_host. The user also requires an SMTP approved sender and its credentials to authenticate with the Email Delivery server to obtain the credential_name. SMTP connection must be configured, and SMTP credentials must be generated and approved.

- For the `msteams` provider, the user requires the Microsoft Teams App and a bot configured in it. The app needs to be published to the org and installed after obtaining approval from the admin from the admin center. The user also requires access for `Files.ReadWrite.All` and `ChannelSettings.Read.All` permission for Graph API from their Azure Portal. To generate the required token, the user requires `bot_id` in the username and `bot_secret` in the password.
- For the `slack` provider, the `username` value can be any valid string and the `password` is the Slack bot token.

### Examples

Send a text message with the `email` provider:

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

BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider        => 'email',
        credential_name => 'EMAIL_CRED',
        message         => 'Subject content',
        params          => json_object('recipient' value
'mark@oracle.com, suresh@oracle.com',
                                       'to_cc'  value
'nicole@oracle.com, jordan@oracle.com',
                                       'to_bcc' value
'manisha@oracle.com',
                                       'subject' value 'Test subject',
                                       'smtp_host' value
'smtp.email.example.com',
                                       'sender'    value
'approver_sender@oracle.com' )
   );
   END;
/
```

Send a text message with the `msteams` provider:

```
BEGIN
    DBMS_CLOUD.CREATE_CREDENTIAL(credential_name => 'TEAMS_CRED',
        username        => 'bot_id',
        password        => 'bot_secret');
END;
/

BEGIN
     DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
        provider        => 'msteams',
        credential_name => 'TEAMS_CRED',
```

```
        message         => 'text from new teams api',
        params          => json_object('channel'value
'C0....08'),'region'value 'india');
END;
/
```

Send a text message with the `slack` provider:

```
BEGIN
   DBMS_CLOUD.CREATE_CREDENTIAL(
     credential_name => 'SLACK_CRED',
     username        => 'SLACK_TOKEN',
     password        => 'password');
END;
/
BEGIN
   DBMS_CLOUD_NOTIFICATION.SEND_MESSAGE(
      provider          => 'slack',
      credential_name   => 'SLACK_CRED',
      message           => 'Send text from Autonomous Database.',
      params            => json_object('channel' value 'C0....08'));
END;
/
```

# DBMS_CLOUD_AI Package

The `DBMS_CLOUD_AI` package facilitates and configures the translation of natural language prompts to SQL statements.

**Prerequisites**

- Oracle Public Cloud
- Exadata Cloud@Customer

## Oracle Public Cloud

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 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 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_AI Subprograms

This section covers the `DBMS_CLOUD_AI` subprograms provided with Autonomous Database.

> **Note:**
>
> You can use the DBMS_CLOUD_AI package only with Autonomous DatabaseAutonomous Database versions 19.22 or later.

| Subprogram | Description |
|---|---|
| CREATE_PROFILE Procedure | This procedure new AI profile for translating natural language prompts to SQL statements. |
| DISABLE_PROFILE Procedure | This procedure disables an AI profile in the current database. |
| DROP_PROFILE Procedure | This procedure drops an existing AI profile. |
| ENABLE_PROFILE Procedure | This procedure enables an AI profile to use in the current database. |
| GENERATE Function | This function generates a SQL statement using AI to translate. |
| SET_ATTRIBUTE Procedure | This procedure sets AI profile attributes. |
| SET_PROFILE Procedure | This procedure sets AI profile for the current database. |

# CREATE_PROFILE Procedure

The procedure creates a new AI profile for translating natural language prompts to SQL statement.

### 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.<br>This is a mandatory parameter. |
| `attributes` | Profile attributes in JSON format. See AI Profile Attributes for more details.<br>The default value is NULL. |
| `status` | Status of the profile.<br>The default value is enable. |
| `description` | Description for the AI profile.<br>The default value is NULL. |

### Example

```
BEGIN
     DBMS_CLOUD_AI.CREATE_PROFILE(
          profile_name    => 'OpenAI,
          attributes      => JSON_OBJECT('provider' value 'openai',
                                         'credential_name' value
'openai_cred'),
          description     => 'AI profile to use OpenAI for SQL
translation'
     );
END;
/
```

## DROP_PROFILE Procedure

The procedure drops an existing AI profile. If the profile does not exist, then the procedure throws an error.

### Syntax

```
DBMS_CLOUD_AI.DROP_PROFILE(
       profile_name        IN   VARCHAR2,
       force               IN   BOOLEAN DEFAULT FALSE
 );
```

### Parameters

| Parameter | Description |
|---|---|
| `profile_name` | Name of the AI profile |
| `force` | If `TRUE`, then the procedure ignores errors if AI profile does not exist.<br>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<br>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.<br>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.<br>This parameter is mandatory. |
| `attribute_name` | Name of the AI profile attribute<br>This parameter is mandatory. |
| `attribute_value` | Value of the profile attribute.<br>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 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.<br>This parameter is mandatory. |

### Example

```
BEGIN
     DBMS_CLOUD_AI.SET_PROFILE(
       profile_name    => 'OPENAI',
     );
END;
/
```

## GENERATE Function

This function provides AI translation in a stateless manner. With your existing AI profile, you can use this function to perform the supported actions such as `showsql`, `narrate`, or `chat`. The default action is `showsql`.
Overriding some or all of the profile attributes is also possible using this function.

### Syntax

```
DBMS_CLOUD_AI.GENERATE(
    prompt          IN  CLOB,
    profile_name    IN  VARCHAR2 DEFAULT NULL,
    action          IN  VARCHAR2 DEFAULT NULL,
    attributes      IN  CLOB     DEFAULT NULL
) RETURN CLOB;
```

### Parameters

| Parameter | Description |
|---|---|
| `prompt` | Natural language prompt to translate using AI.<br>The prompt can include `SELECT AI <action>` as the prefix. The action can also be supplied separately as an "action" parameter. The *action* supplied in prompt overrides the "action" parameter. Default action is `showsql`.<br>This parameter is mandatory. |

| Parameter | Description |
|---|---|
| profile_name | Name of the AI profile. This parameter is optional if an AI profile is already set in the session using DBMS_CLOUD_AI.SET_PROFILE.<br>The default value is NULL.<br>The following conditions apply:<br>• If a profile is set in the current session, the user may omit profile_name argument in the DBMS_CLOUD_AI.GENERATE function.<br>• If the profile_name argument is supplied in the DBMS_CLOUD_AI.GENERATE function, it overrides any value set in the session using the DBMS_CLOUD_AI.SET_PROFILE procedure.<br>• If there is no profile set in the session using the DBMS_CLOUD_AI.SET_PROFILE procedure, the profile_name argument must be supplied in the DBMS_CLOUD_AI.GENERATE function. |

**Note:**

For Database Actions, you can either specify profile_name argument in DBMS_CLOUD_AI.GENERATE or you can run two steps as a PL/SQL script: DBMS_CLOUD_AI.SET_PROFILE and DBMS_CLOUD_AI.GENERATE.

```
EXEC
DBMS_CLOUD_AI.set_profile('OPE
NAI');

------------------------------
-----------------
SELECT
DBMS_CLOUD_AI.GENERATE(prompt
      => 'how many customers',

profile_name => 'OPENAI',

action       => 'showsql')
FROM dual;

------------------------------
-----------------
SELECT
DBMS_CLOUD_AI.GENERATE(prompt
      => 'how many customers',

profile_name => 'OPENAI',

action       => 'narrate')
FROM dual;
------------------------------
```

| Parameter | Description |
|---|---|
| | ```
------------
SELECT
DBMS_CLOUD_AI.GENERATE(prompt
      => 'what is oracle
autonomous database',

profile_name => 'OPENAI',

action       => 'chat')
FROM dual;
```<br><br>See Executing SQL Statements in the Code Editor for more information. |
| `action` | Action for translating natural prompt using AI. The supported actions include `showsql` (default), `narrate`, and `chat`. Descriptions of actions are included in Use AI Keywords to Enter Prompts.<br><br>**Note:**<br><br>This function does not support the `runsql` action. If you supply the `runsql` action, it returns the following error:<br><br>```
ORA-20000: runsql action is
not supported by generate
function ORA-06512: at
"C##CLOUD$SERVICE.DBMS_CLOUD",
 line xxxx
ORA-06512: at
"C##CLOUD$SERVICE.DBMS_CLOUD_A
I", line 2696 ORA-06512: at
line x
``` |
| `attributes` | Override specific AI profile attributes by supplying attributes in JSON format. See Profile Attributes for more details. |

### Examples

The following examples illustrate `showsql`, `narrate`, and `chat` actions that can be used with the `DBMS_CLOUD_AI.GENERATE` function.

An example with `showsql` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt       => 'how many customers',
                              profile_name => 'OPENAI',
```

```
                                action       => 'showsql')
FROM dual;
```

An example with `narrate` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt       => 'how many customers',
                              profile_name => 'OPENAI',
                              action       => 'narrate')
FROM dual;
```

An example with `chat` action is as follows:

```
SELECT DBMS_CLOUD_AI.GENERATE(prompt       => 'what is oracle autonomous
database',
                              profile_name => 'OPENAI',
                              action       => 'chat')
FROM dual;
```

# Profile Attributes

Attributes of an AI profile help to manage and configure the behavior of the AI profile. Some attributes are optional and have a default value.

**Attributes**

| Attribute Name | Description |
|---|---|
| `azure_deployment_name` | Name of the Azure OpenAI Service deployed model. The name can only include alphanumeric characters, underscore character (_) and a hyphen (-) character. The name cannot end with an underscore (_) or a hyphen (-). To know how to get the `azure_deployment_name`, see Create and deploy an Azure OpenAI Service resource. |
| `azure_resource_name` | Name of the Azure OpenAI Service resource. The resource name can only include alphanumeric characters and hyphens, and can't start or end with a hyphen. To know how to get the `azure_resource_name`, see Create and deploy an Azure OpenAI Service resource. |
| `comments` | Include column comments in the metadata used for translating natural language prompts using AI. `BOOLEAN` datatype is supported. The valid values are `TRUE` or `FALSE` for a string with `VARCHAR2` datatype. |

> **Note:**
>
> Boolean values are not applicable in the `DBMS_CLOUD_AI.SET_ATTRIBUTE` procedure when setting a single attribute because `attribute_value` parameter is of `CLOB` datatype.

| Attribute Name | Description |
|---|---|
| credential_name | The name of the credential to access the AI provider APIs.<br>Credential using bearer tokens can be created by using the provider name as the user name and bearer token as the password.<br>This is a mandatory attribute. See CREATE_CREDENTIAL Procedure. |
| max_tokens | Denotes the number of tokens to predict per generation. Default is 1024. See BPE Tokens for more details. |
| model | The name of the AI model being used to generate translation.<br>Supported models:<br>• 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<br>• For Cohere: command (default), command-nightly (experimental), command-light, and command-light-nightly (experimental). Smaller, "light" models are faster, while larger models will perform better. Custom models can also be supplied with their full ID.<br>**Note:**<br>This parameter is not used for Azure as the model is determined when you create your deployment in the Azure OpenAI Service portal. |
| 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.<br>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.<br>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.<br>**Format:**<br>`[ {"owner": "SH", "name": "SALES", {"owner": "TEST_USER"} ]` |
| provider | AI provider for the AI profile.<br>Supported providers:<br>• openai<br>• cohere<br>• azure<br>This is a mandatory attribute. |

| Attribute Name | Description |
|---|---|
| `stop_tokens` | The generated text will be terminated at the beginning of the earliest stop sequence. Sequence will be incorporated into the text. The attribute value must be a valid array of string values in JSON format. `stop_tokens` takes a JSON array as input. To learn more about stop tokens or stop sequences, see OpenAI or Cohere documentation. |
| `temperature` | Sampling from generation models incorporates randomness, so that the same prompt may yield different outputs each time you hit "generate". Temperature is a non-negative float number used to tune the degree of randomness. Lower temperatures mean less random generations. See Temperature for more details. This parameter is applicable to all the supported service providers. |

The following example is using Cohere as the provider and displays custom profile attributes:

```
BEGIN
  DBMS_CLOUD_AI.CREATE_PROFILE(
     profile_name => 'COHERE',
     attributes =>
      '{"provider": "cohere",
        "credential_name": "COHERE_CRED",
        "object_list": [{"owner": "ADMIN"}],
        "max_tokens":512,
        "stop_tokens": [";"],
        "model": "command-nightly",
        "temperature": 0.5,
        "comments": true
       }');
END;
/
```

# Troubleshooting Autonomous Database on Dedicated Exadata Infrastructure

Use the following sections to help troubleshoot problems you have with Oracle Autonomous 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 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 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 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 here: APPEND_HOST_ACE Procedure

## 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

# API Reference for Autonomous Database on Dedicated Exadata Infrastructure

You can use API to view and manage the different infrastructure resources of an Autonomous 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.

| Operation | REST API Endpoint | Required IAM Permissions |
|---|---|---|
| Create an Exadata Infrastructure resource | CreateCloudExadataInfrastructure | CLOUD_EXADATA_INFRASTRUCTURE_CREATE<br>VNIC_CREATE<br>SUBNET_ATTACH<br>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<br>VNIC_DELETE<br>SUBNET_DETACH<br>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 |

| Operation | REST API Endpoint | Required IAM Permissions |
|---|---|---|
| Download an Exadata Infrastructure resource configuration file | DownloadExadataInfrastructureConfigFile | EXADATA_INFRASTRUCTURE_CONTENT_READ |
| 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.

| 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 |
| 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:** 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 |

| Operation | REST API Endpoint | Required IAM Permissions |
|---|---|---|
| Change the backup retention policy of an Autonomous Container Database<br>or<br>Edit the maintenance preferences of an Autonomous Container Database | UpdateAutonomousContainerDatabase | AUTONOMOUS_CONTAINER_DATABASE_UPDATE |
| Restart an Autonomous Container Database | RestartAutonomousContainerDatabase | AUTONOMOUS_CONTAINER_DATABASE_UPDATE |
| 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 | APPLIES TO: Exadata Cloud@Customer only<br>AUTONOMOUS_CONTAINER_DATABASE_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_INSPECT |
| Terminate an Autonomous Container Database | TerminateAutonomousContainerDatabase | **On Oracle Public Cloud:**<br>CLOUD_EXADATA_INFRASTRUCTURE_UPDATE and CLOUD_AUTONOMOUS_VM_CLUSTER_UPDATE and AUTONOMOUS_CONTAINER_DATABASE_DELETE<br>**On Exadata Cloud@Customer:**<br>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 |

# API to Manage Autonomous Databases

The following table lists the REST API endpoints to manage Autonomous Databases.

| Operation | REST API Endpoint | Required IAM Permissions |
|---|---|---|
| Create an Autonomous Database<br>or<br>Clone an Autonomous Database | CreateAutonomousDatabase | AUTONOMOUS_DATABASE_CREATE |
| View a list of Autonomous Databases | ListAutonomousDatabases | AUTONOMOUS_DATABASE_INSPECT |

| Operation | REST API Endpoint | Required IAM Permissions |
|---|---|---|
| View details of an Autonomous Database | GetAutonomousDatabase | AUTONOMOUS_DATABASE_INSPECT |
| View a list of character sets supported by Autonomous Database. | ListAutonomousDatabaseCharacterSets | None |
| Set the password of an Autonomous Database's ADMIN user<br>or<br>Scale the CPU core count or storage of an Autonomous Database<br>or<br>Enable or disable auto scaling for an Autonomous Database | UpdateAutonomousDatabase | AUTONOMOUS_DATABASE_UPDATE |
| Move an Autonomous 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 Database key | RotateAutonomousDatabaseEncryptionKey | AUTONOMOUS_DATABASE_UPDATE |
| Start an Autonomous Database | StartAutonomousDatabase | AUTONOMOUS_DATABASE_UPDATE |
| Stop an Autonomous Database | StopAutonomousDatabase | AUTONOMOUS_DATABASE_UPDATE |
| Restart an Autonomous Database | RestartAutonomousDatabase | AUTONOMOUS_DATABASE_UPDATE |
| Terminate an Autonomous Database | DeleteAutonomousDatabase | AUTONOMOUS_DATABASE_DELETE |
| Restore an Autonomous Database | RestoreAutonomousDatabase | AUTONOMOUS_DB_BACKUP_CONTENT_READ and AUTONOMOUS_DATABASE_CONTENT_WRITE |
| Back up an Autonomous Database manually | CreateAutonomousDatabaseBackup | AUTONOMOUS_DB_BACKUP_CREATE and AUTONOMOUS_DATABASE_CONTENT_READ |
| View the list of Autonomous Database backups | ListAutonomousDatabaseBackups | AUTONOMOUS_DB_BACKUP_INSPECT |
| Get the details of an Autonomous 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 | REST API Endpoint | Required IAM Permissions |
|---|---|---|
| View the Autonomous Data Guard Association with an Autonomous Container Database. | GetAutonomousContainerDatabaseDataguardAssociation | AUTONOMOUS_CONTAINER_DATABASE_INSPECT |
| List the Autonomous Container Databases enabled with Autonomous Data Guard associated with the specified Autonomous Container Database or Autonomous Database. | ListAutonomousContainerDatabaseDataguardAssociations | AUTONOMOUS_CONTAINER_DATABASE_INSPECT |
| Reinstate the Disabled Standby in an Autonomous Data Guard Configuration. | ReinstateAutonomousContainerDatabaseDataguardAssociation | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE |
| Switch Roles in an Autonomous Data Guard Configuration. | SwitchoverAutonomousContainerDatabaseDataguardAssociation | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE |
| Fail Over to the Standby in an Autonomous Data Guard Configuration. | FailoverAutonomousContainerDatabaseDataguardAssociation | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE |
| Enable or Disable Automatic Failover in an Autonomous Data Guard Configuration. | UpdateAutonomousContainerDatabaseDataguardAssociation | AUTONOMOUS_CONTAINER_DATABASE_INSPECT and AUTONOMOUS_CONTAINER_DATABASE_UPDATE |
| Get an Autonomous Data Guard-enabled database associated with the specified Autonomous Database. | GetAutonomousDatabaseDataguardAssociation | AUTONOMOUS_CONTAINER_DATABASE_INSPECT |
| List Autonomous Database data guard associations. | ListAutonomousDatabaseDataguardAssociations | AUTONOMOUS_CONTAINER_DATABASE_INSPECT |
| Enable Autonomous Data Guard on an Autonomous Container Database | CreateAutonomousContainerDatabaseDataguardAssociation | CLOUD_AUTONOMOUS_VM_CLUSTER_INSPECT, AUTONOMOUS_CONTAINER_DATABASE_INSPECT, and AUTONOMOUS_CONTAINER_DATABASE_UPDATE |

For complete details of each REST API endpoint along with examples, see:

- Autonomous Container Database Dataguard Association Reference
- Autonomous Database Dataguard Association Reference

## 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 |
| Lists the Autonomous Virtual Machines in the specified Autonomous VM Cluster and Compartment. | ListAutonomousVirtualMachines | AUTONOMOUS_VIRTUAL_MACHINE_INSPECT |

# Events for Autonomous Database on Dedicated Exadata Infrastructure

APPLIES TO: Oracle Public Cloud only

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 Database on Oracle Cloud.

> **Tip:**
>
> Refer to Oracle Exadata Cloud@Customer Events for listings of the operations that generate events for Autonomous Database on Exadata Cloud@Customer.

Autonomous 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 Database infrastructure resources emit:

- Cloud Exadata Infrastructure Event Types
- Cloud Autonomous VM Cluster Event Types
- Autonomous Container Database Event Types
- Autonomous Database Event Types
- Autonomous Data Guard Event Types

# Cloud Exadata Infrastructure Event Types

Review the list of event types that Exadata Infrastructure instances emit.

### Cloud Exadata Infrastructure Event Types

| Friendly Name | Event Type |
|---|---|
| Cloud Exadata Infrastructure - Create Begin | `com.oraclecloud.databaseservice.createcloudexadatainfrastructure.begin` |
| Cloud Exadata Infrastructure - Create End | `com.oraclecloud.databaseservice.createcloudexadatainfrastructure.end` |
| Cloud Exadata Infrastructure - Change Compartment Begin | `com.oraclecloud.databaseservice.changecloudexadatainfrastructurecompartment.begin` |
| Cloud Exadata Infrastructure - Change Compartment End | `com.oraclecloud.databaseservice.changecloudexadatainfrastructurecompartment.end` |
| Cloud Exadata Infrastructure - Critical<br>See Cloud Exadata Infrastructure Critical and Information Event Types for details. | `com.oraclecloud.databaseservice.cloudexadatainfrastructure.critical` |
| Cloud Exadata Infrastructure - Delete Begin | `com.oraclecloud.databaseservice.deletecloudexadatainfrastructure.begin` |
| Cloud Exadata Infrastructure - Delete End | `com.oraclecloud.databaseservice.deletecloudexadatainfrastructure.end` |
| Cloud Exadata Infrastructure – Information<br>See Cloud Exadata Infrastructure Critical and Information Event Types for details. | `com.oraclecloud.databaseservice.cloudexadatainfrastructure.information` |
| Cloud Exadata Infrastructure – Maintenance Begin | `com.oraclecloud.databaseservice.cloudexadatainfrastructuremaintenance.begin` |
| Cloud Exadata Infrastructure – Maintenance End | `com.oraclecloud.databaseservice.cloudexadatainfrastructuremaintenance.end` |
| Cloud Exadata Infrastructure – Maintenance Reminder | `com.oraclecloud.databaseservice.cloudexadatainfrastructuremaintenancereminder` |
| Cloud Exadata Infrastructure – Maintenance Scheduled | `com.oraclecloud.databaseservice.cloudexadatainfrastructuremaintenancescheduled` |
| Cloud Exadata Infrastructure - Update Begin | `com.oraclecloud.databaseservice.updatecloudexadatainfrastructure.begin` |
| Cloud Exadata Infrastructure - Update End | `com.oraclecloud.databaseservice.updatecloudexadatainfrastructure.end` |

### Example 7-1 Cloud Exadata Infrastructure Event

This is a reference event for a Cloud Exadata Infrastructure resource:

```
{
  "cloudEventsVersion": "0.1",
  "eventId": "<unique_ID>",
  "eventType":
"com.oraclecloud.databaseservice.cloudexadatainfrastructuremaintenance.end",
  "source": "DatabaseService",
```

```
  "eventTypeVersion": "1.0",
  "eventTime": "2019-06-27T21:16:04.000Z",
  "contentType": "application/json",
  "extensions": {
    "compartmentId": "ocid1.compartment.oc1.<unique_ID>"
  },
  "data": {
    "compartmentId": "ocid1.compartment.oc1.<unique_ID>",
    "compartmentName": "example_name",
    "resourceName": "my_exadata_infrastructure",
    "resourceId": "ocid1.dbsystem.oc1.eu-frankfurt-1.<unique_ID>", ,
    "availabilityDomain": "tXPJ:EU-FRANKFURT-1-AD-3",
    "freeFormTags": {
      "Department": "Finance"
    },
    "definedTags": {
      "Operations": {
        "CostCenter": "42"
      }
    },
    "additionalDetails" : {
"subnetId" : "ocid1.subnet.oc1.eu-frankfurt-1.<unique_ID>",
"lifecycleState" : "MAINTENANCE_IN_PROGRESS",
"sshPublicKeys" : "...",
"cpuCoreCount" : 32,
"version" : "19.2.8.0.0.191119",
"nsgIds" : "null",
"backupSubnetId" : "ocid1.subnet.oc1.eu-frankfurt-1.<unique_ID>",
"licenseType" : "BRING_YOUR_OWN_LICENSE",
"dataStoragePercentage" : 80,
"patchHistoryEntries" : "null",
"lifecycleMessage" : "The underlying infrastructure of this system
(cell storage) is being updated and this will not impact database
availability.",
"exadataIormConfig" : "ExadataIormConfigCache(lifecycleState=DISABLED,
lifeCycleDetails=null, objective=Auto,
dbPlans=[DbIormConfigCache(dbName=default, share=null,
flashCacheLimit=null), DbIormConfigCache(dbName=<my_database1>,
share=null, flashCacheLimit=null),
DbIormConfigCache(dbName=<my_database2>, share=null,
flashCacheLimit=null), DbIormConfigCache(dbName=<my_database3>,
share=null, flashCacheLimit=null),
DbIormConfigCache(dbName=<my_database4>, share=null,
flashCacheLimit=null), DbIormConfigCache(dbName=<my_database5>,
share=null, flashCacheLimit=null),
DbIormConfigCache(dbName=<my_database6>, share=null,
flashCacheLimit=null), DbIormConfigCache(dbName=<my_database7>,
share=null, flashCacheLimit=null),
DbIormConfigCache(dbName=<my_database8>, share=null,
flashCacheLimit=null), DbIormConfigCache(dbName=<my_database9>,
share=null, flashCacheLimit=null),
DbIormConfigCache(dbName=<my_database10>, share=null,
flashCacheLimit=null), DbIormConfigCache(dbName=<my_database11>,
share=null, flashCacheLimit=null)], undoData=null)"
}
```

```
},
"eventID" : "<unique_ID>",
"extensions" : {
"compartmentId" : "ocid1.compartment.oc1.<unique_ID>"
}
}
```

### Cloud Exadata Infrastructure Critical and Information Event Types

Cloud Exadata Infrastructure resources emit "critical" and "information" event types. These event types allow you to receive notifications when your infrastructure resource needs urgent attention ("critical" events) or notifications for events that are not critical but which you may want to monitor ("information" events). The eventType values for these events are the following:

| Friendly Name | Event Type |
|---|---|
| Cloud Exadata Infrastructure - Critical | `com.oraclecloud.databaseservice.exadata infrastructure.critical` |
| Cloud Exadata Infrastructure - Information | `com.oraclecloud.databaseservice.exadata infrastructure.information` |

These events use the `additionalDetails` section of the event message to provide specific details about what is happening within the infrastructure resource emitting the event. In the `additionalDetails` section, the `eventName` field provides the name of the critical or information event. *(Note that some fields in the example that follows have been omitted for brevity.)*

```
{
  "eventType" :
"com.oraclecloud.databaseservice.exadatainfrastructure.critical",
  ....
  "data" : {
   ....
     "additionalDetails" : {
      ....
      "description" : "SQL statement terminated by Oracle Database Resource
Manager due to excessive consumption of CPU and/or I/O. The execution plan
associated with the terminated SQL stmt is quarantined. Please find the sql
identifier in sqlId field of this JSON payload. This feature protects an
Oracle database from performance degradation. Please review the SQL
statement. You can see the statement using the following commands: \"set
serveroutput off\", \"select sql_id, sql_text from v$sqltext where sql_id
=<sqlId>\", \"set serveroutput on\"",
      "component" : "storage",
      "infrastructureType" : "exadata",
      "eventName" : "CloudExadataInfrastructure.SqlQuarantine",
      "quarantineMode" : "\"FULL Quarantine\""
       ....
    }
  },
  "eventID" : "<unique_ID>",
  ....
```

```
  }
}
```

For full-length example critical and information events, see Example: Critical Event for Cloud Exadata Infrastructure Resource and Example: Information Event for Cloud Exadata Infrastructure Resource.

In the tables below, you can read about the conditions and operations that trigger "critical" and "information" events. Each condition or operation is identified by a unique `eventName` value.

### Critical Events for Cloud Exadata Infrastructure

| Critical Event - EventName | Description |
|---|---|
| `CloudExadataInfrastructure.SqlQuarantine` | SQL statement is terminated by Oracle Database Resource Manager due to excessive consumption of CPU and/or I/O. The execution plan associated with the terminated SQL statement is quarantined. Please find the sql identifier in `sqlId` field of this JSON payload. This feature protects an Oracle database from performance degradation. Please review the SQL statement. You can see the statement using the following commands:<br><br>`\"set serveroutput off\"`<br>`\"select sql_id, sql_text from v$sqltext where sql_id =<sqlId>\"`<br>`\"set serveroutput on\"` |

### Informational Events for Cloud Exadata Infrastructure

| Information Event - EventName | Description |
|---|---|
| `CloudExadataInfrastructure.FlashDiskFailure` | Flash Disk Failure has been detected. This is being investigated by Oracle Exadata team and the disk will be replaced if needed. No action needed from the customer. |

### Example: Critical Event for Cloud Exadata Infrastructure Resource

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 SQL statement that was terminated by Oracle Database Resource Manager because it was consuming excessive CPU or I/O resources. The `eventName` and `description` fields within the `additionalDetails` section provide information regarding the critical situation:

```
{
  "eventType" :
"com.oraclecloud.databaseservice.exadatainfrastructure.critical",
  "cloudEventsVersion" : "0.1",
  "eventTypeVersion" : "2.0",
  "source" : "Exadata Storage",
```

```
  "eventTime" : "2021-07-30T04:53:18Z",
  "contentType" : "application/json",
  "data" : {
    "compartmentId" : "ocid1.tenancy.oc1.<unique_ID>",
    "compartmentName" : "example_name",
    "resourceName" : "my_exadata_resource",
    "resourceId" : "ocid1.dbsystem.oc1.phx.<unique_ID>",
    "availabilityDomain" : "phx-ad-2",
     "additionalDetails" : {
      "serviceType" : "exacs",
      "sqlID" : "gnwfm1jgqcfuu",
      "systemId" : "ocid1.dbsystem.oc1.eu-frankfurt-1.<unique_ID>",
      "creationTime" : "2021-05-14T13:29:28+00:00",
      "dbUniqueID" : "1558836122",
      "quarantineType" : "SQLID",
      "dbUniqueName" : "AB0503_FRA1S6",
      "description" : "SQL statement terminated by Oracle Database Resource
Manager due to excessive consumption of CPU and/or I/O. The execution plan
associated with the terminated SQL stmt is quarantined. Please find the sql
identifier in sqlId field of this JSON payload. This feature protects an
Oracle database from performance degradation. Please review the SQL
statement. You can see the statement using the following commands: \"set
serveroutput off\", \"select sql_id, sql_text from v$sqltext where sql_id
=<sqlId>\", \"set serveroutput on\"",
      "quarantineReason" : "Manual",
      "asmClusterName" : "None",
      "component" : "storage",
      "infrastructureType" : "exadata",
      "name" : "143",
      "eventName" : "CloudExadataInfrastructure.SqlQuarantine",
      "comment" : "None",
      "quarantineMode" : "\"FULL Quarantine\"",
      "rpmVersion" : "OSS_20.1.8.0.0_LINUX.X64_210317",
      "cellsrvChecksum" : "14f73eb107dc1be0bde757267e931991",
      "quarantinePlan" : "SYSTEM"
    }
  },
  "eventID" : "<unique_ID>",
  "extensions" : {
    "compartmentId" : "ocid1.tenancy.oc1.<unique_ID>"
  }
}
```


### Example: Information Event for Cloud Exadata Infrastructure Resource

In the following example of an "information" event, you can see within the `additionalDetails` section of the event message that this particular message concerns a flash disk failure that is being investigated by the Oracle Exadata operations team. The `eventName` and `description` fields within the `additionalDetails` section provide information regarding the event:


```
{
  "eventType" :
"com.oraclecloud.databaseservice.exadatainfrastructure.information",
  "cloudEventsVersion" : "0.1",
```

```
  "eventTypeVersion" : "2.0",
  "source" : "Exadata Storage",
  "eventTime" : "2021-12-17T19:14:42Z",
  "contentType" : "application/json",
  "data" : {
    "compartmentId" :
"ocid1.tenancy.oc1..aaaaaaaao3lj36x6lwxyvc4wausjouca7pwyjfwb5ebsq5emrpq
lql2gj5iq",
    "compartmentName" : "intexadatateam",
    "resourceId" :
"ocid1.dbsystem.oc1.phx.abyhqljt5y3taezn7ug445fzwlngjfszbedxlcbctw45ykk
axyzc5isxoula",
    "availabilityDomain" : "phx-ad-2",
    "additionalDetails" : {
      "serviceType" : "exacs",
      "component" : "storage",
      "systemId" :
"ocid1.dbsystem.oc1.phx.abyhqljt5y3taezn7ug445fzwlngjfszbedxlcbctw45ykk
axyzc5isxoula",
      "infrastructureType" : "exadata",
      "description" : "Flash Disk Failure has been detected. This is
being investigated by Oracle Exadata team and the disk will be
replaced if needed. No action needed from the customer.",
      "eventName" : "CloudExadataInfrastructure.FlashDiskFailure",
      "FLASH_1_1" : "S2T7NA0HC01251  failed",
      "otto-ingestion-time" : "2021-12-17T19:14:43.205Z",
      "otto-send-EventService-time" : "2021-12-17T19:14:44.198Z"
    }
  },
  "eventID" : "30130ab4-42fa-4285-93a7-47e49522c698",
  "extensions" : {
    "compartmentId" :
"ocid1.tenancy.oc1..aaaaaaaao3lj36x6lwxyvc4wausjouca7pwyjfwb5ebsq5emrpq
lql2gj5iq"
  }
}
```


# Cloud Autonomous Exadata VM Cluster Event Types

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.autonomouscloudvmcluster.changecompartment.begin` |
| Autonomous Cloud VM Cluster - Change Compartment End | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.changecompartment.end` |
| Autonomous Cloud VM Cluster - Create Begin | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.create.begin` |

| Friendly Name | Event Type |
|---|---|
| Autonomous Cloud VM Cluster - Create End | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.create.end` |
| Autonomous Cloud VM Cluster - Critical<br>See Cloud Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details. | `com.oraclecloud.databaseservice.autonomous.cloudautonomousvmcluster.critical` |
| Autonomous Cloud VM Cluster - Information<br>See Cloud Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details. | `com.oraclecloud.databaseservice.autonomous.cloudautonomousvmcluster.information` |
| Autonomous Cloud VM Cluster - Maintenance Begin | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.maintenance.begin` |
| Autonomous Cloud VM Cluster - Maintenance End | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.maintenance.end` |
| Autonomous Cloud VM Cluster - Maintenance Reminder | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.maintenance.reminder` |
| Autonomous Cloud VM Cluster - Maintenance Scheduled | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.maintenance.scheduled` |
| Autonomous Cloud VM Cluster - Terminate Begin | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.terminate.begin` |
| Autonomous Cloud VM Cluster - Terminate End | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.terminate.end` |
| Autonomous Cloud VM Cluster - Update Begin | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.update.begin` |
| Autonomous Cloud VM Cluster - Update End | `com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.update.end` |
| Autonomous Cloud VM Cluster - Warning<br>See Cloud Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types for details. | `com.oraclecloud.databaseservice.autonomous.cloudautonomousvmcluster.warning` |

### Example 7-2 Cloud Autonomous Exadata VM Cluster Event


```
{
  "cloudEventsVersion": "0.1",
  "eventID": "60600c06-d6a7-4e85-b56a-1de3e6042f57",
  "eventType":
"com.oraclecloud.databaseservice.autonomous.autonomouscloudvmcluster.update.b
egin",
  "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"
    }
  }
}
```


### Cloud Autonomous Exadata VM Cluster Critical, Warning, and Information Event Types

Cloud 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:

| 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 Cloud Autonomous Exadata VM Cluster

| Event | Description |
|---|---|
| `sslcertificate.expired` | This event is generated when the SSL Certificate expires. All the Autonomous 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. |

### Warning Events for 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 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. |

| Event | Description |
|---|---|
| autonomous.operations.issue.resolved.information | Autonomous Dedicated Operations has resolved an issue with your Autonomous Exadata VM Cluster. No further action is required. |

# Autonomous Container Database Event Types

Review the list of event types that Autonomous Container Databases emit.

| Friendly Name | Event Type |
|---|---|
| Autonomous Container Database - Change Compartment | com.oraclecloud.databaseservice.changeautonomouscontainerdatabasecompartment.begin |
| 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 |
| 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 - 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 - Restart Begin | com.oraclecloud.databaseservice.restartautonomouscontainerdatabase.begin |
| Autonomous Container Database - Restart End | com.oraclecloud.databaseservice.restartautonomouscontainerdatabase.end |
| Autonomous Container Database - Restore Begin | com.oraclecloud.databaseservice.autonomous.container.database.restore.begin |

| Friendly Name | Event Type |
|---|---|
| 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 - 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 7-3 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. |

| Event | Description |
|---|---|
| backup.destination.error | Autonomous Dedicated Operations has detected one of the following critical issues with your backup destination:<br>• Your Network File Storage server is out of space.<br>• Incorrect permissions are set on your Network File Storage server.<br>• Your Zero Data Loss Recovery Appliance server is out of space. |

### 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 | Generated when Network File Storage backups consume at least 80% of space in the backup destination. |

### 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 | Your Autonomous Container Database's backup destination is back to a healthy state. |

## Autonomous Database Event Types

Review the list of event types that Autonomous Databases emit.

### Autonomous Database Event Types

| Friendly Name | Event Type |
|---|---|
| Autonomous Database - Automatic Backup Begin | com.oraclecloud.databaseservice.automaticbackupautonomousdatabase.begin |

| Friendly Name | Event Type |
|---|---|
| Autonomous Database - Automatic Backup End | com.oraclecloud.databaseservice.automaticbackupautonomousdatabase.end |
| Autonomous Database - Backup Update – Begin | com.oraclecloud.databaseservice.autonomous.database.backup.updateautonomousdatabasebackup.begin |
| Autonomous Database - Backup Update – End | com.oraclecloud.databaseservice.autonomous.database.backup.updateautonomousdatabasebackup.end |
| Autonomous Database - Change Compartment Begin | com.oraclecloud.databaseservice.changeautonomousdatabasecompartment.begin |
| Autonomous Database - Change Compartment End | com.oraclecloud.databaseservice.changeautonomousdatabasecompartment.end |
| Autonomous Database - Change Database Name Begin | com.oraclecloud.databaseservice.changeautonomousdatabasename.begin |
| Autonomous Database - Change Database Name End | com.oraclecloud.databaseservice.changeautonomousdatabasename.end |
| Autonomous Database - Create Backup Begin | com.oraclecloud.databaseservice.autonomous.database.backup.begin |
| Autonomous Database - Create Backup End | com.oraclecloud.databaseservice.autonomous.database.backup.end |
| Autonomous Database - Create Begin | com.oraclecloud.databaseservice.autonomous.database.instance.create.begin |
| Autonomous Database - Create End | com.oraclecloud.databaseservice.autonomous.database.instance.create.end |
| Autonomous Database - Critical<br>See Autonomous Database Critical, Warning, and Information Event Types for details. | com.oraclecloud.databaseservice.autonomous.database.critical |
| Autonomous Database - Information<br>See Autonomous Database Critical, Warning, and Information Event Types for details. | com.oraclecloud.databaseservice.autonomous.database.information |
| Autonomous Database - Deregister Autonomous Database with Data Safe Begin | com.oraclecloud.databaseservice.deregisterautonomousdatabasedatasafe.begin |
| Autonomous Database - Deregister Autonomous Database with Data Safe End | com.oraclecloud.databaseservice.deregisterautonomousdatabasedatasafe.end |
| Autonomous Database - Generate Wallet | com.oraclecloud.databaseservice.generateautonomousdatabasewallet |
| Autonomous Database - Long-term Backup Will Expire In 24 Hours | com.oraclecloud.databaseservice.autonomous.database.backup.longtermbackupexpiresinaday.reminder |
| Autonomous Database - Long-term Backup Will Expire In 30 Days | com.oraclecloud.databaseservice.autonomous.database.backup.longtermbackupexpiresinamonth.reminder |
| Autonomous Database - Long-term Backup Will Expire In 90 Days | com.oraclecloud.databaseservice.autonomous.database.backup.longtermbackupexpiresinthreemonths.reminder |
| Autonomous Database - Register Autonomous Database with Data Safe Begin | com.oraclecloud.databaseservice.registerautonomousdatabasedatasafe.begin |

| Friendly Name | Event Type |
|---|---|
| Autonomous Database - Register Autonomous Database with Data Safe End | com.oraclecloud.databaseservice.registerautonomousdatabasedatasafe.end |
| Autonomous Database - Restart Begin | com.oraclecloud.databaseservice.restartautonomousdatabase.begin |
| Autonomous Database - Restart End | com.oraclecloud.databaseservice.restartautonomousdatabase.end |
| Autonomous Database - Restore Begin | com.oraclecloud.databaseservice.autonomous.database.restore.begin |
| Autonomous Database - Restore End | com.oraclecloud.databaseservice.autonomous.database.restore.end |
| Autonomous Database - Rotate Encryption Key Begin | com.oraclecloud.databaseservice.rotateautonomousdatabaseencryptionkey.begin |
| Autonomous Database - Rotate Encryption Key End | com.oraclecloud.databaseservice.rotateautonomousdatabaseencryptionkey.end |
| Autonomous Database - Start Begin | com.oraclecloud.databaseservice.startautonomousdatabase.begin |
| Autonomous Database - Start End | com.oraclecloud.databaseservice.startautonomousdatabase.end |
| Autonomous Database - Stop Begin | com.oraclecloud.databaseservice.stopautonomousdatabase.begin |
| Autonomous Database - Stop End | com.oraclecloud.databaseservice.stopautonomousdatabase.end |
| Autonomous Database - Terminate Begin | com.oraclecloud.databaseservice.deleteautonomousdatabase.begin |
| Autonomous Database - Terminate End | com.oraclecloud.databaseservice.deleteautonomousdatabase.end |
| Autonomous Database - Update Begin | com.oraclecloud.databaseservice.updateautonomousdatabase.begin |
| Autonomous Database - Update End | com.oraclecloud.databaseservice.updateautonomousdatabase.end |
| Autonomous Database - Warning<br>See Autonomous Database Critical, Warning, and Information Event Types for details. | com.oraclecloud.databaseservice.autonomous.database.warning |

### Example 7-4 Autonomous Database Event

This is a reference event for an Autonomous 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": "<Data Warehouse | Transaction Processing>",
      "autonomousDatabaseType": "<Dedicated Infrastructure | Shared
Infrastructure>"
    }
  }
 }
```

### Autonomous Database Critical, Warning, and Information Event Types

Autonomous 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 Database emitting the event. The `eventType` values for each of these events are listed below:

| Friendly Name | Event Type |
|---|---|
| Autonomous Database - Critical | `com.oraclecloud.databaseservice.autonomous.database.critical` |
| Autonomous Database - Warning | `com.oraclecloud.databaseservice.autonomous.database.warning` |
| Autonomous Database - Information | `com.oraclecloud.databaseservice.autonomous.database.information` |

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 Database

| Event | Description |
|---|---|
| `adminpassword.expired` | If the ADMIN password expires and is no longer usable, Autonomous Database reports an event that specifies that the ADMIN user password has expired and must be reset. |
| `autonomous.operations.issue.detected.critical` | Autonomous Dedicated Operations has detected a critical issue with your Autonomous 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 Database. No further action is required. |
| `wallet.expired` | Your Autonomous Database wallet has expired. |
| `wallet.rotated` | Your Autonomous Database wallet has been rotated and must be downloaded. |

### Warning Events for Autonomous Database

| Event | Description |
|---|---|
| adminpasswordexpiry.warning | Generated when Autonomous 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 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 Database. No further action is required. |
| storagequotalimit.warning | Generated when the storage quota usage exceeds 90%. |
| walletexpiry.warning | Generated when Autonomous 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 Database**

| Event | Description |
|---|---|
| autonomous.operations.issue.detected.information | Autonomous Dedicated Operations has detected an issue with your Autonomous 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 Database. No further action is required. |

# Autonomous Data Guard Event Types

Review the list of event types that Autonomous Data Guard associations emit for dedicated Exadata infrastructure.

| Friendly Name | Event Type |
|---|---|
| Autonomous Data Guard Association - Create Begin | com.oraclecloud.databaseservice.createautonomousdataguardassociation.begin |
| Autonomous Data Guard Association - Create End | com.oraclecloud.databaseservice.createautonomousdataguardassociation.end |
| Autonomous Data Guard Association - Critical See Autonomous Data Guard Association Critical Event Types. | com.oraclecloud.databaseservice.autonomous.container.database.dataguardassociation.critical |
| Autonomous Data Guard Association - Failover Begin | com.oraclecloud.databaseservice.failoverautonomousdataguardassociation.begin |

| Friendly Name | Event Type |
|---|---|
| Autonomous Data Guard Association - Failover End | `com.oraclecloud.databaseservice.failoverautonomousdataguardassociation.end` |
| Autonomous Data Guard Association - Reinstate Begin | `com.oraclecloud.databaseservice.reinstateautonomousdataguardassociation.begin` |
| Autonomous Data Guard Association - Reinstate End | `com.oraclecloud.databaseservice.reinstateautonomousdataguardassociation.end` |
| Autonomous Data Guard Association - Switchover Begin | `com.oraclecloud.databaseservice.switchoverautonomousdataguardassociation.begin` |
| Autonomous Data Guard Association - Switchover End | `com.oraclecloud.databaseservice.switchoverautonomousdataguardassociation.end` |

### Autonomous Data Guard Association Critical Event Types

Autonomous Data Guard associations emit critical event types. These event types allow you to receive notifications when your Autonomous Data Guard association needs urgent attention.

The `eventType` value for these events is `com.oraclecloud.databaseservice.autonomous.container.database.dataguardassociation.critical`

These events use the `additionalDetails` section of the event message to provide specific details about what is happening with the Autonomous Data Guard association emitting the event. In the `additionalDetails` section, the `eventName` field provides the name of the critical event.

In the table below, you can read about the conditions and operations that trigger critical events. Each condition or operation is identified by a unique `eventName` value.

### Critical Events for Autonomous Data Guard Associations

| Event | Description |
|---|---|
| `rolechange.primary` | Generated when an Autonomous Database changes its role from standby to primary. This role change could result from manual failover, switchover, or automatic failover. |
| `rolechange.standby` | Generated when an Autonomous Database changes its role from primary to standby. This role change could result from manual failover, switchover, or automatic failover. |

# Availability Service Level Agreements (SLAs)

This topic describes the Service Level Agreements (SLAs) and Service Level Objectives (SLOs) for Oracle Autonomous Database on Dedicated Exadata Infrastructure.

Oracle Autonomous Database runs on the Oracle Exadata Cloud infrastructure (Oracle Public Cloud and Oracle Exadata Cloud@Customer), leveraging Oracle's Maximum Availability Architecture (MAA). Autonomous 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. See Maximum Availability Architecture and Autonomous Database Cloud for more details about Oracle MAA.

### Uptime

The following table outlines the Service Level Agreement (SLA) and Service Level Objective (SLO) for Oracle Autonomous Database on Dedicated Exadata Infrastructure.

**Table 7-1 Uptime SLAs/SLOs**

| Service | Type | Uptime (without Autonomous Data Guard) | Uptime (with Autonomous Data Guard) |
|---|---|---|---|
| Autonomous Database on Dedicated Exadata Infrastructure (Oracle Public Cloud deployments) | Service Level Agreement (SLA) | 99.95%<br>A maximum of 22 minutes of downtime per month. | 99.995%<br>A maximum of 132 seconds of downtime per month. |
| Autonomous Database on Exadata Cloud@Customer | Service Level Objective (SLO) | 99.95%<br>A maximum of 22 minutes of downtime per month. | 99.995%<br>A maximum of 132 seconds of downtime per month. |
| Autonomous Database for Developers<br>(Both Oracle Public Cloud and Exadata Cloud@Customer deployments) | Service Level Objective (SLO) | 99.5% | Not applicable<br>Autonomous 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 Database on Dedicated Exadata Infrastructure without Autonomous Data Guard and with Autonomous Data Guard.

**Table 7-2 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:<br>• Exadata cluster network topology failures<br>• Storage (disk and flash) failures<br>• Database instance failures<br>• Database server failures<br>• Periodic software and hardware maintenance updates | Near-zero | Zero |
| Events that require restoring from backup because the standby database does not exist:<br>• Data corruptions<br>• Full database failures<br>• Complete storage failures<br>• Availability domain (AD) for multi-AD regions | Minutes to hours<br>(without Autonomous Data Guard) | 15 minutes<br>(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.<br>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 7-3 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:<br>• Exadata cluster network fabric failures<br>• Storage (disk and flash) failures<br>• Database instance failures<br>• Database server failures<br>• Periodic software and hardware maintenance updates | Zero or Near Zero | Zero |
| Events that require failover to the standby database using Autonomous Data Guard, including:<br>• 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)<br>• Full database failures<br>• Complete storage failures<br>• 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.<br>• Near zero for maximum performance protection mode (uses asynchronous redo transport). Most commonly used for cross-region standby databases. |

# Certify Applications for Compatibility Between Autonomous Database Serverless and Dedicated

Considering that your application is already certified to use Autonomous Database Serverless, the following table helps you evaluate if your application can be certified to use Autonomous Database on Dedicated Exadata Infrastructure too.

The table below lists the options and features that are either:

- Currently available only in Autonomous Database Serverless, or
- Has *differences in functionality* with Autonomous Database on Dedicated Exadata Infrastructure deployments both in Public Cloud and Cloud@Customer.

> **Note:**
>
> There are also features only supported in Autonomous Database on Dedicated Exadata Infrastructure deployment and not supported in Autonomous Database Serverless deployments. However, as the presumption is that you are evaluating to certify your application on both the deployments, starting with Autonomous Database Serverless deployments, we are not discussing them here.

**Table 7-4 Autonomous Database Serverless Vs Autonomous Database on Dedicated Exadata Infrastructure Features**

| Feature/Option | Autonomous Database Serverless | Autonomous Database on Dedicated Exadata Infrastructure |
|---|---|---|
| **Database In-Memory** | Not supported currently. | Supported.<br>Refer to Database In-Memory for more information. |
| **Database Link Outbound Support** | `DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK` (TCPS only)<br>Only public IP, public hostname, or using an Oracle Database Gateway is supported. See DBMS_CLOUD_ADMIN.CREATE_DATABASE_LINK. | `CREATE_DATABASE_LINK` (TCP only)<br>Public and private connections are supported. See the *CREATE DATABASE LINK* entry in Limitations on the Use of SQL Commands. |
| **Database Renaming** | Supported. | Not supported currently. |
| **Database Resident Connection Pooling** | Use Database Resident Connection Pooling with Autonomous Database Serverless | Not supported currently. |
| **`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`<br>See List of Initialization Parameters that can be Modified in Autonomous Database Serverless. | For differences with Autonomous Database Serverless, see Modifying Database Initialization Parameters. |
| **Oracle Application Express (APEX) Support** | See Restrictions and Limitations for Oracle Application Express with Autonomous Database. | For differences with Autonomous Database Serverless, see Limitations on the Use of Oracle Application Express. |
| **Oracle Cloud Infrastructure PL/SQL SDK API** | Supported. | Not supported currently. |

**Table 7-4 (Cont.) Autonomous Database Serverless Vs Autonomous Database on Dedicated Exadata Infrastructure Features**

| Feature/Option | Autonomous Database Serverless | Autonomous Database on Dedicated Exadata Infrastructure |
|---|---|---|
| **Oracle Spatial and Graph** | This feature is provided through Oracle Maps Cloud Service.<br>• Geocoder<br>• Routing Engine<br>• Spatial Visualization<br>See Oracle Spatial Limitations with Autonomous Database Serverless. | For differences with Autonomous Database Serverless, see Limitations on the Use of Oracle Spatial and Graph. |
| **Oracle Machine Learning (OML)** | Supports:<br>• OML4SQL on all database versions<br>• OML4Py<br>• OML4R<br>• OML Notebooks (SQL, PL/SQL, Python, R, markdown)<br>• OML AutoML UI<br>• OML Services<br>• Oracle Data Miner<br>For OML product documentation, refer to Oracle Machine Learning on Oracle Help Center. | Supports:<br>• OML4SQL on all database versions.<br>• Oracle Data Miner<br>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 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.<br>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 Database Serverless, see Notes on the Use of Database PL/SQL Packages. |

**Table 7-4 (Cont.) Autonomous Database Serverless Vs Autonomous Database on Dedicated Exadata Infrastructure Features**

| Feature/Option | Autonomous Database Serverless | Autonomous Database on Dedicated Exadata Infrastructure |
|---|---|---|
| `UTL_SMTP` | Supported. | Supported. For differences with Autonomous Database Serverless, see Notes on the Use of Database PL/SQL Packages. |

# Feature Parity Between Oracle Public Cloud and Exadata Cloud@Customer

Autonomous 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 Databases, there are a few differences among them.

### Feature Comparison

The following table lists the Autonomous Database features exclusive to either Oracle Public Cloud or Exadata Cloud@Customer deployments.

| Feature | Oracle Public Cloud | Exadata Cloud@Customer |
|---|---|---|
| Authenticate and Authorize Microsoft Azure Active Directory Users for Autonomous Database | ✔ | |
| Change the license type on an Autonomous VM Cluster | Supported via service request (SR) in My Oracle Support. See Create a Service Request in My Oracle Support for instructions. | ✔ |
| Connect Python applications using the python-oracledb driver | | ✔ |
| Deploy Autonomous Database to X10M Exadata Database Machine Systems | | ✔ |
| Deploy Autonomous 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. |
| Identity and Access Management (IAM) authentication with Autonomous Database | ✔ | |
| Use Database Management Service to monitor databases | ✔ | |

| Feature | Oracle Public Cloud | Exadata Cloud@Customer |
|---|---|---|
| Use Operations Insights on Autonomous Database | ✔ | |

# 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 Database on Dedicated Exadata Infrastructure.

> **Note:**
>
> See Resource Limits for the maximum and recommended resource limits for Autonomous Database on Dedicated Exadata Infrastructure on Oracle Public Cloud and Exadata Cloud@Customer.

## Exadata Base Systems

**Autonomous 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 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 X10M Systems

### Autonomous Database on Exadata Cloud@Customer

For Autonomous 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 Database on Dedicated Exadata Infrastructure deployed on Exadata Cloud@Customer.

| Property | Elastic X10M (min config) | Elastic X10M-L (min config) | Elastic X10M-XL (min config) |
|---|---|---|---|
| Minimum number of X10M DB servers | 2 | 2 | 2 |
| Minimum number of X10M Storage Servers | 3 | 3 | 3 |
| Total number of usable DB node OCPUs (assuming 2 cores per host) | 380 per server | 380 per server | 380 per server |
| Total memory available for VMs (GB) | 2780 | 4180 | 5600 |
| Maximum Local Storage per Database Server (GB) | 2243 | 2243 | 2243 |
| Max /u02 capacity (GB) per VM | 900 | 900 | 900 |
| VM Image overhead per VM (GB) | 184 | 184 | 184 |
| Max local storage (/u02 + VM Image capacity) (GB) per Infrastructure | 2243 | 2243 | 2243 |
| Total number of VM clusters* | 8 | 8 | 8 |
| Minimum CPUs per VM | 2 | 2 | 2 |
| Minimum memory per VM (GB) | 30 | 30 | 30 |
| Min Local File System Storage per VM (GB) | 60 | 60 | 60 |
| Min Exadata Storage per VM Cluster (TB) | 2 | 2 | 2 |
| Total Usable ASM storage capacity (TB) per Cell | 80 | 80 | 80 |
| Exadata RMDA Memory (XRMEM) in TB | 1.25 per storage server | 1.25 per storage server | 1.25 per storage server |

* A maximum of 16 clusters can be created on a single system. For more information, see Scaling Local Storage.

# Exadata X9M-2 Systems

### Autonomous Database on Oracle Public Cloud

For Autonomous 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 Requesting 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 |
|---|---|
| Number of Compute Nodes | 2 |
| Total Maximum Number of Enabled CPU Cores | 252 |
| Total RAM Capacity (GB) | 2,780 |
| Persistent Memory (TB) | 4.5 |
| Number of Exadata Storage Servers | 3 |
| Maximum Database Size, No Local Backup (TB) | 152 |

For more details, see the Dedicated Autonomous Database X9M data sheet.

### Autonomous Database on Exadata Cloud@Customer

For Autonomous 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 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 |
|---|---|---|---|---|
| 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 |
| 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 Database on Oracle Public Cloud

For Autonomous 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 Requesting 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 |
|---|---|
| Number of Compute Nodes | 2 |
| Total Maximum Number of Enabled CPU Cores | 100 |
| Total RAM Capacity (GB) | 2,780 |
| Persistent Memory (TB) | 4.5 |

| Property | Default Provisioning |
|---|---|
| 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 Database on Exadata Cloud@Customer**

For Autonomous 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 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 |
|---|---|---|---|---|
| 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 |
| 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 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 |

| Specification | Exadata X8-2 Base Rack | Exadata X8-2 Quarter Rack | Exadata X8-2 Half Rack | Exadata X8-2 Full Rack |
|---|---|---|---|---|
| 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 Database on Dedicated Exadata Infrastructure.

> **Note:**
>
> For Oracle Autonomous 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 Database on Dedicated Exadata Infrastructure

You can provision more Autonomous 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 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 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 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 Transaction Processing workloads.
- The `low` services for Autonomous Data Warehouse workloads.

The resources allocated to databases provisioned with OCPU overprovisioning are calculated proportionally based on the number of OCPUs chosen. For example, an Autonomous Database with 0.3 OCPUs will get memory and concurrent statement allocations that is 30% of the allocation for a single OCPU Autonomous Database. See Predefined Database Service Names for Autonomous Databases for more details.

When creating a new Autonomous Database instance or cloning an existing Autonomous Database instance or its backup:

- CPU overprovisioning is not supported for databases using ECPUs. The minimum ECPU value for an Autonomous 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 Database from Backup for instructions on cloning from a backup.

When scaling an Autonomous 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 Transaction Processing workloads and the `low` services for the Autonomous Data Warehouse workloads. Refer to Add CPU or Storage Resources to an Autonomous 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 toRemove CPU or Storage Resources from an Autonomous Database for instructions on scaling down your database.

Auto-scaling enables an Autonomous 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)

# Previous Feature Announcements

## New Feature Announcements (2023)

Announcements for the noteworthy changes made to Autonomous 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.<br>Refer to the following for more information:<br>• Create an Autonomous Container Database<br>• 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.<br>This ability lets you scale up or scale down an AVMC by modifying the following resources:<br>• Number of CPUs per VM<br>• Number of Autonomous Container Databases (ACDs)<br>• Database storage<br>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 Database to provide standard, hardended security configurations for your Autonomous VMs (Client VMs).<br>Refer to Configuration Management in Autonomous Database for details. |

| Feature | Description |
| --- | --- |
| 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.<br>For more information, see Enable Autonomous Data Guard on Autonomous Container Database. |
| 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 Database instance. This enables you to compare workloads between an on-premises database or other cloud service database and an Autonomous Database.<br>You can also capture workload from an Autonomous Database on Dedicated Exadata Infrastructure and replay it on the same or any other Autonomous Database. This can be useful in scenarios where you want to test a patch or upgrade before applying it on your production database.<br>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 Database on Dedicated Exadata Infrastructure for complete information about the key security features supported by Autonomous Database on Dedicated Exadata Infrastructure.<br>This document comes with a clickable graphic representing the building blocks for the key security features of the Autonomous Database. You can also click the block you are interested in exploring further. |

| Feature | Description |
| --- | --- |
| 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.<br>Updating the time-zone file would require a complete downtime for the ACD and the associated Autonomous Databases. The downtime is dependent on the amount of data that is time-zone sensitive.<br>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. |
| 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.<br>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 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 Database on Dedicated Exadata Infrastructure for details. |
| New compute model: ECPU | You can configure your Autonomous Database resources using a new compute type called ECPUs.<br>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.<br>For more information about the compute models supported by Autonomous Database, see Compute Models in Autonomous Database. |

| Feature | Description |
| --- | --- |
| 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.<br>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.<br>Understanding and knowing these details can help you optimize your resource allocation to Autonomous Databases and efficiently forecast your capacity needs.<br>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 Database deployments on Exadata Cloud@Customer, you can now deploy to X10M Exadata Database Machine systems.<br>For details about the capacities and characteristics of the X10M systems supported by Autonomous Database, see Exadata X10M Systems. |
| Limit Operator's Access to a Specific Customer-Approved Autonomous Container Database (ACD) | Autonomous 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.<br>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.<br>Refer to Auditing Oracle Operator Activities to know more about about Oracle operator access control. |

| Feature | Description |
| --- | --- |
| Parallel Replicat in Integrated Mode | Autonomous 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.<br><br>**Note:**<br>Parallel Replicat is not supported for overprovisioned Autonomous Databases, that is, databases provisioned with less than 1 OCPU. |
| Documentation Update: Troubleshooting Guide | To help troubleshoot problems you might encounter with Oracle Autonomous Database on Dedicated Exadata Infrastructure, Troubleshooting Autonomous 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.<br><br>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.<br><br>Understanding and knowing these details can help you optimize your resource allocation to Autonomous Databases and efficiently forecast your capacity needs.<br><br>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. |

| Feature | Description |
| --- | --- |
| Database In-Memory Column Store support for Autonomous Database on Dedicated Exadata Infrastructure | You can enable Database In-memory for your Autonomous Database by allocating a percentage of its System Global Area (SGA) to the In-Memory column store (IM column store).<br>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.<br>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.<br>Please see Database In-Memory for the requirements and guidelines to use this feature with Autonomous Database.<br>Refer to Enable or Disable Database In-Memory or Create an Autonomous Database on Dedicated Exadata Infrastructure for detailed instructions. |

## July 2023

| Feature | Description |
| --- | --- |
| Cross region Autonomous Data Guard configuration for Autonomous 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.<br>This choice provides the highest level of protection against disasters, including a catastrophic loss of external network connectivity or power to an entire region.<br>For more details about Autonomous Data Guard and related concepts, refer to Protect Critical Databases from Failures and Disasters Using Autonomous Data Guard. |

| Feature | Description |
| --- | --- |
| Clone a database instance or its backup across regions and Exadata Infrastructure resources. | On Exadata Cloud@Customer deployments, you can:<br>• Clone an Autonomous 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 Databases with Autonomous Data Guard, you can clone only from the primary database.<br>• Create a new Autonomous Database from a backup on the same or another Exadata Infrastructure in the same or a different region. In the case of Autonomous 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.<br>For step-by-step instructions, see:<br>• Clone an Autonomous Database Instance<br>• Clone Autonomous Database from Backup |

## June 2023

| Feature | Description |
| --- | --- |
| Documentation update to consolidate the performance tuning and monitoring features supported by Autonomous Database on Dedicated Exadata Infrastructure. | The existing documentation related to the high-performance features supported by Autonomous Database on Dedicated Exadata Infrastructure is improvised to provide comprehensive information in a single article.<br>The new article has:<br>• 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 Database.<br>• A detailed explanation of each feature and tool with cross-references to the related how-to guides, wherever appropriate.<br>Refer to High Performance Features in Autonomous Database on Dedicated Exadata Infrastructure for more information. |
| Base Shape support for Autonomous Database on Dedicated Exadata Infrastructure on Oracle Public Cloud. | Oracle Autonomous Database on Dedicated Exadata Infrastructure comes in different infrastructure shapes to support workloads of different sizes. In this release, the capability of Autonomous Database on Dedicated Exadata Infrastructure has been extended to support Base shape.<br>See Resource Limits and Characteristics of Infrastructure Shapes for more information. |

| Feature | Description |
| --- | --- |
| Use `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` to enable an external authentication scheme. | You can use `DBMS_CLOUD_ADMIN.ENABLE_EXTERNAL_AUTHENTICATION` to enable Kerberos, Microsoft Azure AD, or Identity and Access Management (IAM) Authentication as an external authentication scheme for your Autonomous Database.<br>Refer to the following topics for detailed instructions:<br>• Enable Kerberos Authentication on Autonomous Database<br>• Configure the Oracle Autonomous Database for Microsoft Azure AD Integration<br>• Change External Identity Providers on Autonomous Database |
| 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.<br>The following procedures support specifying files in a directory with the `file_uri_list` parameter:<br>• `DBMS_CLOUD.COPY_COLLECTION`<br>• `DBMS_CLOUD.COPY_DATA`<br>• `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`<br>The following procedures support specifying files in a directory with the `partitioning_clause` parameter:<br>• `DBMS_CLOUD.CREATE_EXTERNAL_TABLE`<br>• `DBMS_CLOUD.CREATE_HYBRID_PART_TABLE`<br>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 Database. You can also create external tables for Avro, Parquet or ORC format data files.<br>Refer to the following topics for more details:<br>• DBMS_CLOUD Avro, ORC, and Parquet Support<br>• COPY_DATA Procedure for Avro, ORC, or Parquet Files<br>• CREATE_EXTERNAL_TABLE Procedure for Avro, ORC, or Parquet Files |

# May 2023

| Feature | Description |
| --- | --- |
| Compartment quota support for Autonomous Database on Exadata Cloud@Customer | Autonomous 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.<br>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 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.<br>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.<br>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 Database on Dedicated Exadata Infrastructure for more details.<br>**Note:**<br>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 Database on Dedicated Exadata Infrastructure | The existing documentation explaining the High-Performance features supported by Autonomous 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 Database, referencing the related how-to guides.<br>For more details, see High Performance Features in Autonomous Database. |

| Feature | Description |
| --- | --- |
| 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.<br>Refer to Manage Security Certificates for an Autonomous Exadata VM Cluster Resource for more details. |
| 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.<br>For more information on using this feature with *Oracle Public Cloud* deployments, see:<br>• Scale CPU cores within an Exadata Cloud Infrastructure instance<br>• Scale CPU cores in an Exadata Cloud Infrastructure cloud VM cluster or DB system<br>• Add compute and storage resources to a flexible cloud Exadata infrastructure resource<br>• Use the API to Manage Exadata Cloud Infrastructure Instance<br>• cloud-exadata-infrastructures<br>For more information on using this feature with *Exadata Cloud@Customer* deployments, see:<br>• Use the Console to Scale Infrastructure Compute and Storage<br>• Use the Console to Download Scale Infrastructure Compute and Storage Configuration File<br>• Use the Console to Activate New Database and Storage Servers<br>• Use the Console to Make Storage Capacity from New Server Available for VM Clusters Consumption<br>• 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 `SYSDATE_AT_DBTIMEZONE` parameter enables special handling in a session for the date and time value returned in calls to `SYSDATE` and `SYSTIMESTAMP`. Depending on the value of `SYSDATE_AT_DBTIMEZONE`, you see either the date and time based on the default Autonomous Database time zone, Coordinated Universal Time (UTC), or based on the time zone that you set in your database.<br>See SYSDATE_AT_DBTIMEZONE for more information. |
| SCAN listener port and TLS authentication mode | On Oracle Public Cloud deployments, you can now:<br>• Choose a SCAN listener port for TLS and non-TLS from a range of available ports.<br>• Choose between one-way TLS and mTLS for database TLS certificates.<br>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 `DBMS_CLOUD_ADMIN` 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.<br>Refer to the following for more information:<br>• Attach External File Storage to Autonomous Database<br>• Attach Network File Storage to Autonomous Database<br>• DBMS_CLOUD_ADMIN Package |
| Base Shape support on Exadata Cloud@Customer deployments. | Oracle Autonomous 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.<br>See Resource Limits and Characteristics of Infrastructure Shapes for more information. |

| Feature | Description |
| --- | --- |
| 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.<br>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.<br>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.<br>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 Database users.<br>For detailed guidance on configuring Kerberos authentication, its prerequisites and limitations, refer to Authenticate Autonomous 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.<br>See Use SQL Tracing on Autonomous Database for more information. |
| Password Profile Support for User-Defined Profiles | You can now create and alter user profiles on Autonomous Database on Dedicated Exadata Infrastructure with a Password Verification Function (PVF).<br>See Manage User Profiles with Autonomous Database for more information. |

| Feature | Description |
|---|---|
| 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.<br>Refer to Connect Power BI and Microsoft Data Tools to Autonomous 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 Database from a backup.<br>Refer to Clone Requirements for more details. |
| Long-Term Backups | You can now create long-term backups on Autonomous 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.<br>Refer to Backup and Restore Autonomous Databases for more details. |
| 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 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.<br>You can now refer to the following sections in Autonomous Database on Dedicated Exadata Infrastructure documentation for assistance with `DBMS_CLOUD` procedures and functions:<br>• DBMS_CLOUD for Access Management<br>• DBMS_CLOUD for Objects and Files<br>• DBMS_CLOUD REST APIs |

| Feature | Description |
|---|---|
| Support to save resource configuration as a stack | While provisioning new Autonomous 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.<br>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.<br>For instructions to use this feature with different Autonomous Database resources, see:<br>• Create an Exadata Infrastructure Resource<br>• Create an Autonomous Exadata VM Cluster<br>• Create an Autonomous Container Database<br>• Create an Autonomous Database |
| Support for Autonomous VM Cluster node subsetting on Autonomous 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.<br>For more details, refer to Create Autonomous Exadata VM Cluster. |
| 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.<br>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.<br>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.<br>See Update Autonomous Data Guard Settings for detailed instructions. |

| Feature | Description |
| --- | --- |
| 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 Database or its immediate predecessor.<br>For example: Suppose the latest Oracle Database version supported by Autonomous 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.<br>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.<br>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 Database on Dedicated Exadata Infrastructure.

## December 2022

| Feature | Description |
| --- | --- |
| Documentation Update: Availability Service Level Agreements (SLAs) for Autonomous Database on Dedicated Exadata Infrastructure | Autonomous 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.<br>Refer to Availability Service Level Agreements (SLAs) for complete details. |

| Feature | Description |
| --- | --- |
| Multiple VM Autonomous Database on Dedicated Exadata Infrastructure | On Oracle Public Cloud deployments, you can now:<br>• Create both Autonomous Exadata VM Clusters and Exadata Database VM Clusters on a common Exadata Infrastructure resource.<br>• Create multiple Autonomous Exadata VM Clusters on an Exadata Infrastructure resource.<br>• Support for multiple VM Clusters lets you:<br>– Schedule separate maintenance runs for each Autonomous VM Cluster on the same Exadata Infrastructure.<br>– Choose different license models for Autonomous Databases on the same Exadata Infrastructure.<br>– Create and test Autonomous Data Guard between Autonomous Exadata VM Clusters on the same Exadata Infrastructure.<br>– Customize compute, storage, and memory of each Autonomous Exadata VM Cluster configuration for the intended workload.<br>– Implement network isolation of Autonomous Exadata VM Clusters to provide security and performance guarantees for specific workloads (Dev-Test, Staging, and Production).<br><br>**Note:**<br>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 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 |

| Feature | Description |
| --- | --- |
| | with filing a support request, see Create a Service Request in My Oracle Support. |
| Documentation Update: IAM Policy Details for Autonomous Database on Dedicated Exadata Infrastructure | Autonomous Database on Dedicated Exadata Infrastructure documentation is updated with the details for writing policies to control access to Autonomous Database resources.<br>Refer to Policy Details for Oracle Autonomous Database for complete details. |

## November 2022

| Feature | Description |
| --- | --- |
| SCAN listener port and TLS authentication mode | On Exadata Cloud@Customer deployments, you can now choose:<br>• A SCAN listener port for TLS and non-TLS from a range of available ports.<br>• Between one-way TLS and mTLS for database TLS certificates.<br>The above choices are available *only* 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.<br>Refer to Connect Python Applications Using mTLS for more details. |

## October 2022

| Feature | Description |
| --- | --- |
| Oracle LogMiner | You can now use LogMiner with your Autonomous Database. LogMiner, which is part of Oracle Database, enables you to query online and archived redo log files through a SQL interface.<br>For limitations on using LogMiner with Autonomous Database, see Limitations on the Use of LogMiner. |

| Feature | Description |
| --- | --- |
| 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 Database.<br>Refer to Oracle API for MongoDB Support for the requirements and procedure involved in using Oracle Database API for MongoDB with an Autonomous Database. |

## August 2022

| Feature | Description |
| --- | --- |
| New events for Autonomous Data Guard associations. | You can now get notified when an Autonomous Database changes its role from standby to primary or primary to standby as a result of manual failover, switchover, or automatic failover.<br>To subscribe to event notifications, see Use Events and Notifications to Watch Resource Activity.<br>Refer to Events for Autonomous Database on Dedicated Exadata Infrastructure for a complete list of events emitted by the Autonomous Database resources. |
| Create a new Autonomous Database instance from backup. | You can now create a new Autonomous Database by cloning from a backup on the same or another Exadata Infrastructure in the same or a different region.<br>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.<br>For more details, see:<br>• Clone Sources<br>• Clone Autonomous Database from Backup |
| Clone a database instance across regions and Exadata Infrastructure resources. | You can now clone an Autonomous Database to an Autonomous Container Database on another Exadata Infrastructure in the same or a different region.<br>For step-by-step instructions, see Clone an Autonomous Database Instance. |
| Fast Ingest Support | You can now optimize the processing of high-frequency, single-row data inserts into Autonomous Database using fast ingest.<br>To learn more about using fast ingest with Autonomous Database, see Limitations on the Use of Fast Ingest. |

## July 2022

| Feature | Description |
|---|---|
| Oracle Operator Access Control for Autonomous Database on dedicated exadata infrastructure. | Operator Access Control is now expanded to provide controls for client virtual machines deployed on Oracle Autonomous Database on dedicated exadata infrastructure.<br>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.<br>See Auditing Oracle Operator Activities for more details. |
| Documentation addition about Operations Insights. | A new article is added to the Autonomous Database documentation that describes how you can use Operations Insights to monitor databases.<br>Refer to Use Operations Insights on Dedicated Autonomous Database 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:<br>• APPLIES TO: Oracle Public Cloud only<br>• APPLIES TO: Exadata Cloud@Customer only<br>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 Database instance using Azure `OAuth2` access tokens.<br>See Authenticate and Authorize Microsoft Azure Active Directory Users for Autonomous Database for more information. |

| Feature | Description |
| --- | --- |
| New events for Cloud Autonomous Exadata VM Cluster and Autonomous Database resources. | You can now get notified when:<br>• An SSL certificate is rotated either manually (using the Oracle Cloud Infrastructure console or API) or automatically on its expiry.<br>• Your Autonomous Database wallet has been rotated and must be downloaded.<br>To subscribe to event notifications, see Use Events and Notifications to Watch Resource Activity .<br>Refer to Events for Autonomous Database on Dedicated Exadata Infrastructure for a complete list of events emitted by Autonomous 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.<br>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.<br>See Create an Exadata Infrastructure Resource for specific details. |
| Documentation addition about monitoring databases with Autonomous Database metrics. | A new article is added to the Autonomous Database documentation that describes how you can use Autonomous Database metrics to monitor databases. The complete list of available metrics for the Autonomous Database is also documented for reference.<br>See Monitor Databases with Autonomous Database Metrics for more details. |

| Feature | Description |
|---|---|
| Character set selection. | You can now choose a character set of your choice from a list of supported character sets while provisioning an Autonomous Database.<br>Even though you are allowed to choose any supported combination of database and national character sets for an Autonomous 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.<br>See Character Set Selection for Dedicated Autonomous Database for more details. |
| Oracle standard tagging for Autonomous 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.<br>For more details on tagging your cloud resources, see Tagging Dedicated Autonomous Database Cloud Resources. |
| 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.<br>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:<br>• Total OCPUs<br>• Available OCPUs<br>• Reclaimable OCPUs<br>• Available Storage in TB<br>• Total Storage in TB<br>• Memory per OCPU<br>• Total number of ACDs that you can provision in this resource in an AVMC resource.<br>• Remaining (available) ACDs that you can provision in an AVMC resource.<br>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.<br>See OCPU Management and Billing for more details. |

| Feature | Description |
| --- | --- |
| 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 Database.<br>Review the following guidelines when selecting a database name while creating or cloning an Autonomous Database:<br>• Maximum 30 characters in length<br>• Contain only alphanumeric characters<br>• Begin with an alphabetic character<br>• Not contain spaces |

## April 2022

| Feature | Description |
| --- | --- |
| Oracle standard tagging for Autonomous 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.<br>For more details on tagging your cloud resources, see Tagging Dedicated Autonomous Database Cloud Resources. |
| Documentation addition listing API to manage the Autonomous Data Guard configuration. | A reference article is added to the Autonomous Database documentation to provide a list of REST API endpoints to manage Autonomous Data Guard configuration.<br>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 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.<br>For more details, see Service Maintenance of Dedicated Autonomous 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 Database documentation to list the minimum required subnet sizes for Autonomous Database deployments on Oracle Cloud to create an Autonomous Exadata VM Cluster successfully.<br>For more details, see IP Requirements for Autonomous Exadata VM Cluster. |

| Feature | Description |
| --- | --- |
| X9M-2 systems available for Autonomous Database on Oracle Public Cloud deployments. | When you create Autonomous Database deployments on Oracle Public Cloud, you can now deploy to X9M-2 Exadata Database Machine systems.<br>For details about the capacities and characteristics of the X9M-2 systems supported by the Autonomous Database, see Exadata X9M-2 Systems. |
| Create a new Autonomous Database instance from backup. | You can now create a new Autonomous Database by cloning from a backup on Exadata Cloud@Customer.<br>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.<br>For more details, see Clone an Autonomous 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.<br>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 Database | With the latest Release Update (RU Version 19.14.0.1), you can now configure Autonomous Database to use Oracle Cloud Infrastructure Identity and Access Management (IAM) authentication and authorization to allow IAM users to access an Autonomous Database with IAM credentials.<br>As of this release, (IAM) authentication and authorization:<br>• Will be enabled on all the newly provisioned Autonomous Container Databases (ACDs) and Autonomous Databases by default.<br>• Can not be used with Autonomous Databases configured with Autonomous Data Guard.<br>See Use Identity and Access Management (IAM) Authentication with Autonomous Database for more details. |

| Feature | Description |
|---|---|
| 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 Database to Cloud Object Store directly.<br>See Export Data From Autonomous Database to Object Store Directly for more details. |
| Multiple VM Autonomous Database | On Exadata Cloud@Customer deployments, you can now:<br>• Create both Autonomous Exadata VM Clusters and Exadata Database VM Clusters on a common Exadata Infrastructure resource.<br>• Create multiple Autonomous Exadata VM Clusters on a single Exadata Infrastructure resource.<br>• Support for multiple VM Clusters lets you:<br>– Schedule separate maintenance runs for each Autonomous VM Cluster on the same Exadata Infrastructure.<br>– Choose different license models for Autonomous Databases on the same Exadata Infrastructure.<br>– Create and test Autonomous Data Guard between Autonomous Exadata VM Clusters on the same Exadata Infrastructure.<br>– Customize compute, storage, and memory of each Autonomous Exadata VM Cluster configuration for the intended workload.<br>– Implement network isolation of Autonomous Exadata VM Clusters to provide security and performance guarantees for specific workloads (Dev-Test, Staging, and Production).<br>See Create an Autonomous Exadata VM Cluster for more 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.<br>See Managing Blockchain Tables for more information. |

| Feature | Description |
| --- | --- |
| Immutable Tables Support | Immutable tables are read-only tables that prevent unauthorized data modifications by insiders and accidental data modifications resulting from human errors.<br>See Managing Immutable Tables for more information. |
| New events for Cloud Autonomous Exadata VM Cluster and Autonomous Database resources. | You can now get notified of new events emitted by the Autonomous Exadata VM Cluster, Autonomous Container Database, and Autonomous Database infrastructure resources.<br>Refer to the following topics for a complete list of events emitted by these infrastructure resources:<br>• Cloud Autonomous Exadata VM Cluster Event Types<br>• Autonomous Container Database Event Types<br>• Autonomous Database Event Types |

## January 2022

| Feature | Description |
| --- | --- |
| Documentation addition to list the event types supported by Autonomous Database on Dedicated Exadata Infrastructure. | A reference article is added to Autonomous Database documentation to provide a list of different event types supported by Autonomous 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.<br>See Event Types for Autonomous Database on Dedicated Exadata Infrastructure for more information. |
| Payment Card Industry Data Security Standard (PCI DSS) compliance | Autonomous 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.<br>See Regulatory Compliance Certification to view the complete list of international and industry-specific compliance standards met by Autonomous Database on Dedicated Exadata Infrastructure. |

| Feature | Description |
| --- | --- |
| Automatic Failover or Fast-Start Failover | Autonomous 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.<br>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 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 Database on Dedicated Exadata Infrastructure. | The infrastructure setup of a Autonomous Database is taking a new shape with the introduction of a new **Autonomous Exadata VM Cluster** infrastructure resource. The primary changes are as follows:<br>• Exadata Infrastructure (EXA) replaces the Autonomous Exadata Infrastructure resource in the existing model.<br>• 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.<br>• Networking, License configuration and Certificate Management shifts from the Exadata Infrastructure resource to the Autonomous Exadata VM Cluster resource.<br>• New events on the Exadata Infrastructure and Autonomous VM Cluster resources.<br>**Further Reference:**<br>• For more details, see Overview of Latest Resource Model.<br>• For a comprehensive comparison with the current resource model, see Comparison with Current Resource Model.<br>• To understand how your existing Autonomous Database infrastructure resources update to the latest resource model, see Updating to Latest Resource Model. |

| Feature | Description |
| --- | --- |
| 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.<br>The details page for the corresponding Autonomous Container Database and its associated Autonomous Databases will now show the latest Key Version OCID and Key History. |

## November 2021

| Feature | Description |
| --- | --- |
| Oracle Operator Access Control for Autonomous Database on Oracle Exadata Cloud@Customer deployments. | Operator Access Control is now expanded to provide controls for client virtual machines deployed on Autonomous Database on Dedicated Exadata Infrastructure on Exadata Cloud@Customer.<br>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.<br>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`.<br>See Sending Mail on Dedicated Autonomous Database for more information. |
| Integration with Database Management Service. | You can now use **Database Management Service** to monitor the health of a single Autonomous Database or a fleet of Autonomous Databases.<br>For more details about the Database Management Service and learning how to enable it for your Autonomous 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 Database on Dedicated Exadata Infrastructure. See Notes on the Use of Database PL/SQL Packages for more details. |

| Feature | Description |
| --- | --- |
| 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.<br>For information about provisioning an Autonomous Container Database with cross-region Dataguard using your own keys, see Create an Autonomous Container Database.<br>For information on using customer-managed keys in Autonomous Database on Dedicated Exadata Infrastructure, see the following topics:<br>• Prepare to Use Customer-Managed Keys<br>• Use Customer-Managed Keys |

## September 2021

| Feature | Description |
| --- | --- |
| X9M-2 systems available on Exadata Cloud@Customer deployments. | When you create Autonomous 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 Database, see Exadata X9M-2 Systems. |
| New tile for SODA on the **Details** page of Autonomous Database. | You can now download SODA drivers for an Autonomous 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 *Shared Server Configuration* for Autonomous Database applications. | Autonomous Database on Dedicated Exadata Infrastructure supports Shared Server Configuration for maintaining legacy applications designed without connection pooling.<br>Autonomous Database documentation is updated to provide information about the Shared Server Configuration and explain how to enable it for Autonomous Database applications. See the Shared Server Configuration section in *Connect for High Performance*. |

## June 2021

| Feature | Description |
|---|---|
| Fractional OCPUs and GB Storage. | You can now create or clone Autonomous 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:<br>• Create a Dedicated Autonomous Database<br>• Clone a Dedicated Autonomous Database<br>You can also scale up from or scale down to a fractional OCPU from the **Details** page of your Autonomous Database. For more information, see:<br>• Add CPU or Storage Resources to a Dedicated Autonomous Database<br>• Remove CPU or Storage Resources from a Dedicated Autonomous Database |
| Customer-managed keys in Oracle Key Vault (OKV) for Autonomous 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 Databases on Exadata Cloud@Customer and use customer-managed keys stored in Oracle Key Vault to secure your critical data.<br>See Use Customer-Managed Keys in Oracle Key Vault for more details. |

## May 2021

| Feature | Description |
|---|---|
| Restrict Access Using a Network Access Control List | You can now define additional security for your Autonomous 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 Database on Dedicated Exadata Infrastructure deployment on Oracle Exadata Cloud@Customer is now extended to Autonomous Database on Dedicated Exadata Infrastructure deployments on *Oracle Cloud*.<br>An ACL allows you to formulate a fine-grained access control policy by limiting your Autonomous Database's network access to specific applications or clients. See Set Access Control List for Dedicated Autonomous Database. |

| Feature | Description |
| --- | --- |
| Oracle Operator Access Control for Autonomous 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.<br>You can now use Oracle Operator Access Control in your Autonomous Database on Oracle Exadata Cloud@Customer deployments.<br>For more information, see Auditing Oracle Operator Activities in *Autonomous Database on Dedicated Exadata Infrastructure Documentation*. |

## 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. |
| Support for Oracle GoldenGate Capture and Replicat | You can now replicate data to an Autonomous Database using Oracle GoldenGate and Oracle GoldenGate Cloud Service. You can capture changes from the Autonomous Database and replicate to any target database or platform that Oracle GoldenGate supports, including other Autonomous Database.<br>See Use Oracle GoldenGate to Replicate Data with Autonomous Database. |
| Oracle Cloud Infrastructure navigation menu changes. | The Oracle Cloud Infrastructure console left navigation menu is upgraded.<br>Some of the improvements include:<br>• Options to customize the menu by pinning your favorite services for quick access or simply picking from your Most Recently Used ones.<br>• Improved categorization of services into logical groups, combined with an inline search of menu items, makes it easier to find services.<br>• 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 Database on Dedicated Exadata Infrastructure deployments on Public Cloud. | When you create dedicated Autonomous DatabaseAutonomous Database deployments on Oracle Public Cloud, you can now deploy to X8M Exadata Database Machine systems.<br>For details about the capacities and characteristics of the X8M systems supported by Autonomous Database, see Characteristics of Infrastructure Shapes. |

| Feature | Description |
| --- | --- |
| 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.<br>For details about using customer -managed keys in Oracle Key Vault, see Use Customer-Managed Keys in 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 Database. Using Database Actions, you can load data and run SQL statements, queries, and scripts in a worksheet.<br>See Connect with Database Actions for more information. |
| 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 Database in *Autonomous Database on Dedicated Exadata Infrastructure Documentation*. |

## 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 Database in *Autonomous Database on Dedicated Exadata Infrastructure Documentation*. |

# New Feature Announcements (2020)

Announcements for the noteworthy changes made to Autonomous 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 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.<br>See Creating Applications with Oracle Application Express on Autonomous Database.<br>**Note:**<br>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 *Autonomous Database on Dedicated Exadata Infrastructure Documentation*. |

## November 2020

| Feature | Description |
| --- | --- |
| Restrict Access Using a Network Access Control List | You can now define additional security to your Autonomous 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 Database's network access to specific applications or clients.<br>See Set Access Control List for Dedicated Autonomous 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.<br>To provide this capability, it creates 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 automatically converts the standby database to the primary database and, as such, it begins servicing your applications.<br>See Configure and Manage Autonomous Data Guard for more information. |
| Manage your own encryption keys | Normally, Oracle Autonomous Database creates manages the encryption keys used to your protect data.<br>Now, Oracle Autonomous Database 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 Database on Oracle Cloud or on Exadata Cloud@Customer.<br>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 Database deployments on Oracle Exadata Cloud@Customer you can now deploy to X8M-2 Exadata Database Machine systems.<br>For details about the capacities and characteristics of the X8M-2 systems supported by Autonomous Database, see Characteristics of Infrastructure Shapes. |

## August 2020

| Feature | Description |
|---|---|
| In-database support for Microsoft Active Directory users | You can configure an Autonomous Database to authenticate and authorize Microsoft Active Directory users. This allows Active Directory users to access the database using their Active Directory credentials.<br>See Use Microsoft Active Directory with Autonomous 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:<br>• View and Manage Scheduled Maintenance of an Autonomous Exadata Infrastructure Resource<br>• 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 Databases.<br>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.<br>For details about using Data Safe with your Autonomous Databases, see Use Data Safe to Assess and Protect Data. |

## June 2020

| Feature | Description |
|---|---|
| Autonomous Database now available on Exadata Cloud@Customer | You can now create dedicated Autonomous Database deployments on Oracle Exadata Cloud@Customer in your data center.<br>For more information, see Introduction to Autonomous Database on Exadata Cloud@Customer. |
| New Performance Hub features | Performance Hub provides these new features for Autonomous Databases:<br>• 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.<br>• The ability to generate and download an AWR (Automatic Workload Repository) report that shows data captured between two points in time (called snapshots).<br>• The ability to change the time zone that Performance Hub uses when selecting timespans and displaying data.<br>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.<br>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 Databases | You can now use these IDE extensions to connect to, browse, and manage Autonomous Databases directly from within the IDE:<br>• Oracle Cloud Infrastructure Toolkit for Eclipse<br>• Oracle Developer Tools for Visual Studio<br>• Oracle Developer Tools for VS Code<br>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 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.<br>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 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.<br>For instructions, see Clone a Dedicated Autonomous Database. |
| Monitor your database using Autonomous 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.<br>For more information, see Use Autonomous 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.<br>For details about the capacities and characteristics of these systems, see Characteristics of Autonomous Exadata Infrastructure Resources. |
| Support for DBMS_CLOUD subprograms | Oracle Autonomous Database 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.<br>For more information, see Load Data from Files in the Cloud, Querying External Data with Autonomous Database and Exporting Data to Other Oracle Databases.<br>For detailed information about all the supported DBMS_CLOUD subprograms, see DBMS_CLOUD Package Reference. |
| Store files in directories in your database | Oracle Autonomous Database 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.<br>For more information, see Creating and Managing Directories. |
| Support for work requests | Oracle Autonomous Database 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 Database.<br>For a list of Oracle Autonomous Database operations that create work requests, see Work Requests Integration. For general information on using work requests in Oracle Cloud Infrastructure, see Work Requests and Work Requests API. |
| Restart your database in one step | Previously, to restart your database you had to stop it, wait for it to stop, and the start it. Now, you can perform a restart in a single operation.<br>For more information, see Restart a Dedicated Database. |
| Support for compartment quotas | Oracle Autonomous Database 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.<br>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.<br>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 Database on Dedicated Exadata Infrastructure in the year 2019.

## September 2019

| Feature | Description |
|---|---|
| Autonomous Data Warehouse supports Dedicated Exadata Infrastructure deployments | When you create an Autonomous Data Warehouse deployment, you can now choose between the Serverless and dedicated deployment options.<br>For information about creating, managing, and using dedicated Autonomous Data Warehouse deployments, see *User's Guide to Oracle Autonomous Data Warehouse Dedicated Deployments*. |
| Clone a database | You can now create a new Autonomous Database by cloning an existing one, choosing to clone either the full database or only the database metadata.<br>For instructions, see Clone a Dedicated Autonomous Database. |
| Monitor databases using Performance Hub | The Oracle Cloud Infrastructure console now includes Performance Hub for Autonomous Databases, which you can use to view real-time and historical performance data.<br>For more information, see Use Performance Hub to Monitor Databases. |
| 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 Database on Dedicated Exadata Infrastructure deployments.<br>For information about specifying network security groups for your Autonomous Exadata Infrastructure resources, see Change the Network Security Groups Assigned to an Autonomous Exadata Infrastructure Resource.<br>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 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.<br>For more information about how the Events service works and how to set up rules and actions, see Overview of Events.<br>For listings of the operations that generate events, see Autonomous 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 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.<br>For information on how to take advantage of this capability, see *Enterprise Manager Cloud Control Administrator's Guide for Oracle Autonomous Databases*. |

## June 2019

A new deployment choice for Autonomous 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 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 Transaction Processing documentation home page for more information.