Oracle® Database Vault

Administrator's Guide

18c
E83684-10
March 2019

Oracle Database Vault Administrator's Guide, 18c

E83684-10

Primary Author: Patricia Huey

Contributors: Taousif Ansari , Tom Best, Sanjay Bharadwaj , Todd Bottger, Ji-won Byun, Ben Chang, Martin Cheng, Chi Ching Chui, Scott Gaetjen, Viksit Gaur, Rishabh Gupta, Lijie Heng, Dominique Jeunot, Peter Knaggs, Suman Kumar, Chon Lee, Rudregowda Mallegowda, Paul Needham, Yi Ouyang, Hozefa Palitanawala, Robert Pang, Gayathri Sairamkrishnan, Vipin Samar, James Spiller, Srividya Tata, Kamal Tbeileh, Saravana Soundararajan, Sudheesh Varma, Peter Wahl, Alan Williams

Contents

2 What to Expect After You Enable Oracle Database Vault

3 Getting Started with Oracle Database Vault

4 Performing Privilege Analysis to Find Privilege Use

5 Configuring Realms

6 Configuring Rule Sets

7 Configuring Command Rules

8 Configuring Factors

9 Configuring Secure Application Roles for Oracle Database Vault

10 Configuring Oracle Database Vault Policies

11 Using Simulation Mode for Logging Realm and Command Rule Activities

12 Integrating Oracle Database Vault with Other Oracle Products

13 DBA Operations in an Oracle Database Vault Environment

14 Oracle Database Vault Schemas, Roles, and Accounts

15 Oracle Database Vault Realm APIs

16 Oracle Database Vault Rule Set APIs

17 Oracle Database Vault Command Rule APIs

18 Oracle Database Vault Factor APIs

19 Oracle Database Vault Secure Application Role APIs

20 Oracle Database Vault Oracle Label Security APIs

21 Oracle Database Vault Utility APIs

22 Oracle Database Vault General Administrative APIs

23 Oracle Database Vault Policy APIs

24 Oracle Database Vault API Reference

25 Oracle Database Vault Data Dictionary Views

A Auditing Oracle Database Vault

B Disabling and Enabling Oracle Database Vault

C Postinstallation Oracle Database Vault Procedures

D Oracle Database Vault Security Guidelines

E Troubleshooting Oracle Database Vault

Index

List of Figures

List of Tables

Preface

Oracle Database Vault Administrator's Guide explains how to configure access control-based security in an Oracle Database environment by using Oracle Database Vault.

Audience

This document is intended for security managers, audit managers, label administrators, and Oracle database administrators (DBAs) who are involved in the configuration of Oracle Database Vault.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information refer to the following documents:

- *Oracle Database Security Guide*
- *Oracle Label Security Administrator’s Guide*
- *Oracle Database Administrator’s Guide*
- *Oracle Database SQL Language Reference*
- *Oracle Multitenant Administrator's Guide*

Oracle Technology Network (OTN)

You can download free release notes, installation documentation, updated versions of this guide, white papers, or other collateral from the Oracle Technology Network (OTN). Visit

http://www.oracle.com/technetwork/database/security/index.html

For security-specific information on OTN, visit

http://www.oracle.com/technetwork/topics/security/whatsnew/index.html

For the latest version of the Oracle documentation, including this guide, visit

http://www.oracle.com/technetwork/documentation/index.html

Oracle Database Vault-Specific Sites

For OTN information specific to Oracle Database Vault, visit

http://www.oracle.com/us/products/database/options/database-vault/overview/index.html

For frequently asked questions about Oracle Database Vault, visit

http://www.oracle.com/technetwork/database/options/oracle-database-vault-external-faq-2032888.pdf

Oracle Store

Printed documentation is available for sale in the Oracle Store at:

https://shop.oracle.com

My Oracle Support (formerly Oracle*MetaLink*)

You can find information about security patches, certifications, and the support knowledge base by visiting My Oracle Support at:

https://support.oracle.com

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Changes in This Release for Oracle Database Vault Administrator's Guide

This preface contains:

Changes in Oracle Database Vault 18c

The following are changes in *Oracle Database Vault Administrator's Guide* for Oracle Database 18c.

- Enhancements to Oracle Database Vault Simulation Mode
  Oracle Database Vault has had a number of changes to simulation mode for this release.
- New Factor Functions
  Starting with this release, four new factor functions are available.
- Ability to Grant Data Pump-Database Vault Authorizations to Roles
  Starting with this release, you can authorize roles to perform Oracle Data Pump operations in an Oracle Database Vault environment.
- Oracle Database Vault Support for Oracle Database Replay
  In this release, you now can perform Oracle Database Replay operations in an Oracle Database Vault environment.

Enhancements to Oracle Database Vault Simulation Mode

Oracle Database Vault has had a number of changes to simulation mode for this release.

- Simulation mode now captures all mandatory realm violations from a SQL statement.
- Simulation mode can capture the full call stack information.
- The default trusted path context factors are now available as separate columns instead of being concatenated together.

Capturing all mandatory realm violations from a SQL statement enables you to see all changes that you may need to make. Otherwise, the first mandatory realm violation may mask other violations that would not be noticed until the original fix is completed and another regression test is run. This enhancement enables faster regression test and application certification.

Seeing the full call stack helps you to identify the original SQL statement that has the violation. In many cases, similar SQL statements are called by different parts of the application. This feature helps an application developer to quickly identify exactly which application code triggered the violation.

Context factors are used to build trusted paths for realms and command rules. There are some commonly used factors for trusted paths, so these were extracted from the single string representation in the last release into their own columns. This enhancement makes it much easier to identify the factors to use in trusted path rule sets.

Related Topics

- About Simulation Mode
  Simulation mode enables you to capture violations in a simulation log instead of blocking SQL execution by Oracle Database Vault realms and command rules.

New Factor Functions

Starting with this release, four new factor functions are available.

The factor functions are as follows:

- `F$DV$_CLIENT_IDENTIFIER`
- `F$DV$_DBLINK_INFO`
- `F$DV$_MODULE`
- `F$PROXY_USER`

Related Topics

- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the `DVF` schema functions when you use the `DBMS_MACADM` PL/SQL package to manage the various factors.

Ability to Grant Data Pump-Database Vault Authorizations to Roles

Starting with this release, you can authorize roles to perform Oracle Data Pump operations in an Oracle Database Vault environment.

In previous releases, you only could grant this authorization to individual users. This enhancement enables administrators to easily manage users through roles for this type of authorization.

Related Topics

- Using Oracle Data Pump with Oracle Database Vault
  Database administrators can authorize Oracle Data Pump users to work in a Database Vault environment.

Oracle Database Vault Support for Oracle Database Replay

In this release, you now can perform Oracle Database Replay operations in an Oracle Database Vault environment.

The following functionality supports this feature:

- `DBMS_MACADM` PL/SQL procedures:
  - `DBMS_MACADM.AUTHORIZE_DBCAPTURE`
  - `DBMS_MACADM.AUTHORIZE_DBREPLAY`
  - `DBMS_MACADM.UNAUTHORIZE_DBCAPTURE`
  - `DBMS_MACADM.UNAUTHORIZE_DBREPLAY`
- Data dictionary views:
  - `DBA_DV_DBCAPTURE_AUTH`
  - `DBA_DV_DBREPLAY_AUTH`

Related Topics

- Using Oracle Database Replay with Oracle Database Vault
  Database administrators can authorize Oracle Database Replay users to work in a Database Vault environment.

Changes in Oracle Database Vault 12c Release 2 (12.2)

The following are changes in *Oracle Database Vault Administrator's Guide* for Oracle Database 12*c* release 2 (12.2):

- New Features
- Deprecated Features

New Features

The following features are new for this release:

- Ability to Create Oracle Database Vault Policies
  An Oracle Database Vault policy groups and manages realms and command rules that have something in common in a single policy.
- Ability to Configure Simulation Mode Protection
  Simulation mode protects Oracle Database Vault security objects so that SQL commands are not blocked, but violations to the security controls are logged.
- Privilege Analysis Enhancements
  Privilege analysis policies now capture more privilege use than in previous releases, find unused privilege grants, and create named capture runs.
- Ability to Create Common Realms and Common Command Rules for Oracle Multitenant
  In a multitenant environment, you now can create common realms and common command rules within the application PDB context.
- ALTER SESSION, ALTER SYSTEM, and CONNECT Command Rule Enhancements
  Command rules now provide more `ALTER SESSION` and `ALTER SYSTEM` functionality, and CONNECT command rule enhancements.
- Enhancements for the Authentication_Method Default Factor
  Starting with this release, the Authentication_Method default factor can be used for external and global user authentication.

- Changed Default Value for SQL92_SECURITY Parameter
  Starting with this release, the default value for the SQL92_SECURITY parameter has changed from FALSE to TRUE.
- Oracle Database Vault Support for Flashback Technology and ILM
  You now can use Oracle Flashback Technology features and Information Lifecycle Management (ILM) features in an Oracle Database Vault-enabled database.
- Support for Rolling Upgrades for Data Guard Logical Standby Databases
  Oracle Data Guard logical standby databases can perform rolling upgrades for Oracle Database Vault-enabled systems using transient logical standby and the DBMS_ROLLING package.

Ability to Create Oracle Database Vault Policies

An Oracle Database Vault policy groups and manages realms and command rules that have something in common in a single policy.

For example, the realms and command rules can have an application or schema in common.

The status of the realms and command rules in this policy can be updated at once, in a single command, instead of performing the update individually on each security object. Also new to this release is the DV_POLICY_OWNER role, which provides the grantee user the authority to perform basic administration tasks. This user can add authorized users to the realms that are included in the policy and change the policy state without being granted the powerful DV_OWNER role.

The following new functionality supports this feature:

- DBMS_MACADM PL/SQL procedures:
  - DBMS_MACADM.ADD_CMD_RULE_TO_POLICY
  - DBMS_MACADM.ADD_OWNER_TO_POLICY
  - DBMS_MACADM.ADD_REALM_TO_POLICY
  - DBMS_MACADM.CREATE_POLICY
  - DBMS_MACADM.DELETE_CMD_RULE_FROM_POLICY
  - DBMS_MACADM.DELETE_OWNER_FROM_POLICY
  - DBMS_MACADM.DELETE_REALM_FROM_POLICY
  - DBMS_MACADM.DROP_POLICY
  - DBMS_MACADM.RENAME_POLICY
  - DBMS_MACADM.UPDATE_POLICY_DESCRIPTION
  - DBMS_MACADM.UPDATE_POLICY_STATE
- Role:
  - DV_POLICY_OWNER
- Data dictionary views:
  - DVSYS.DBA_DV_POLICY
  - DVSYS.DBA_DV_POLICY_OBJECT
  - DVSYS.DBA_DV_POLICY_OWNER

  - DVSYS.POLICY_OWNER_COMMAND_RULE
  - DVSYS.POLICY_OWNER_POLICY
  - DVSYS.POLICY_OWNER_REALM
  - DVSYS.POLICY_OWNER_REALM_AUTH
  - DVSYS.POLICY_OWNER_REALM_OBJECT
  - DVSYS.POLICY_OWNER_RULE
  - DVSYS.POLICY_OWNER_RULE_SET
  - DVSYS.POLICY_OWNER_RULE_SET_RULE

Related Topics

- Configuring Oracle Database Vault Policies
  You can use Oracle Database Vault policies to implement frequently used realm and command rule settings.
- DV_POLICY_OWNER Database Vault Owner Role
  The DV_POLICY_OWNER role enables database users to manage to a limited degree Oracle Database Vault policies.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

Ability to Configure Simulation Mode Protection

Simulation mode protects Oracle Database Vault security objects so that SQL commands are not blocked, but violations to the security controls are logged.

Simulation mode can also be set for the new Oracle Database Vault policy, which then sets it on embedded objects. This status is between the full protection of being enabled and blocking the access of SQL statements and being disabled. When realms and command rules are set to the simulation mode, violations that occur to these security controls are only logged to the simulation log file. They are not enforced or deny access to the user. This enables you to more quickly certify applications with Oracle Database Vault and to move new realms and command rules to production. You can implement a mandatory realm on a test application database schema and then set it to simulation mode. After running the full test application regression test, you then can analyze the simulation mode log to determine which users and which trusted paths to authorize to the test application schema. You can also develop new command rules and migrate them to production in simulation mode for a period of time to determine the impact they would have when they are enabled.

The simulation mode log is available for viewing in the DBA_DV_SIMULATION_LOG data dictionary view. After you have finished testing the realm and command rule, you can clear the contents of the DBA_DV_SIMULATION_LOG view for the next time you need to test Database Vault configurations in a test environment.

The following functionality supports this features:

- New or changed DBMS_MACADM PL/SQL procedures:
  - DBMS_MACADM.CREATE_REALM (changed)
  - DBMS_MACADM.UPDATE_REALM (changed)

  - DBMS_MACADM.CREATE_COMMAND_RULE (changed)
  - DBMS_MACADM.UPDATE_COMMAND_RULE (changed)
  - DBMS_MACADM.DELETE_COMMAND_RULE (changed)
  - DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE (new)
  - DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE (new)
  - DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE (new)
  - DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE (new)
  - DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE (new)
  - DBMS_MACADM.DELETE_SESSION_EVENT_CMD_RULE (new)
  - DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE (new)
  - DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE (new)
  - DBMS_MACADM.DELETE_SYSTEM_EVENT_CMD_RULE (new)
- New data dictionary view and table:
  - DBA_DV_SIMULATION_LOG data dictionary view
  - DVSYS.SIMULATION_LOG$ table

Related Topics

- Using Simulation Mode for Logging Realm and Command Rule Activities
  Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.
- Oracle Database Vault Realm APIs
  The DBMS_MACADM PL/SQL package enables you to configure Oracle Database Vault realms.
- Oracle Database Vault Command Rule APIs
  The DBMS_MACADM PL/SQL package provides procedures for configuring command rules. .
- DBA_DV_SIMULATION_LOG View
  The DBA_DV_SIMULATION_LOG data dictionary view captures simulation log information for realms and command rules that have had simulation mode enabled.

Privilege Analysis Enhancements

Privilege analysis policies now capture more privilege use than in previous releases, find unused privilege grants, and create named capture runs.

- **Additional privilege captures:** You now can create privilege analysis policies that capture compilation privileges that are used for definer's rights and invoker's rights program units, that capture privileges that were used for Code Based Access Control (CBAC) role use, and that capture secure application role use.
- **Unused privilege grants:** The privilege capture reports now indicate which privilege grants were not used by users or roles.
- **Capture runs:** A capture run defines the period of time that a privilege capture takes place. You define the capture run when you enable the policy. You can

create multiple capture runs for use with one policy and then use these capture runs in comparison reports.

Related Topics

- Performing Privilege Analysis to Find Privilege Use
  Privilege analysis dynamically analyzes the privileges and roles that users use and do not use.

Ability to Create Common Realms and Common Command Rules for Oracle Multitenant

In a multitenant environment, you now can create common realms and common command rules within the application PDB context.

The benefit of creating common realms and command rules (that is, in the application root) is that you can manage them from a central location in a multitenant environment, rather than in individual pluggable databases (PDBs). Realms for the application root common objects must be configured in the application PDB or root. Local realms and local command rules can still be implemented on individual PDBs over and above any common realms and common command rules.

Common realms can only be created on common objects in the application root. You cannot create common realms in the CDB root. However, you can create common command rules in either the application root or the CDB root. A common command rule in the application root applies to its associated PDBs. Common command rules that are in the CDB root will apply to all PDBs in the CDB environment. When you create a common object in the application root and in the CDB root, you must synchronize it so that it is visible in the individual PDBs. To synchronize an object in the application root, you use the `ALTER PLUGGABLE DATABASE APPLICATION` statement with the `SYNC` clause.

The following functionality supports this feature:

- New or changed `DBMS_MACADM` PL/SQL procedures:
  - All realm-related `DBMS_MACADM` PL/SQL procedures
  - All command rule-related `DBMS_MACADM` PL/SQL procedures
  - `DBMS_MACADM.ADD_CMD_RULE_TO_POLICY` (new)
  - `DBMS_MACADM.DELETE_CMD_RULE_FROM_POLICY` (new)
- New or changed data dictionary views:
  - `DBA_DV_COMMAND_RULE`
  - `DVSYS.DBA_DV_POLICY_OBJECT` (new)
  - `DBA_DV_REALM`
  - `DVSYS.DV$REALM`
  - `DBA_DV_REALM_AUTH`
  - `DBA_DV_REALM_OBJECT`
  - `DBA_DV_RULE_SET`
  - `DBA_DV_RULE_SET_RULE`

Related Topics

- Realms in a Multitenant Environment
  In a multitenant environment, you can create a realm to protect common objects in the application root.
- Command Rules in a Multitenant Environment
  In a multitenant environment, you can create common and local command rules in either the CDB root or the application root.
- Oracle Database Vault Realm APIs
  The `DBMS_MACADM` PL/SQL package enables you to configure Oracle Database Vault realms.
- Oracle Database Vault Command Rule APIs
  The `DBMS_MACADM` PL/SQL package provides procedures for configuring command rules. .
- Oracle Database Vault Policy APIs
  You can use the `DBMS_MACADM` PL/SQL package to manage Oracle Database Vault policies.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

ALTER SESSION, ALTER SYSTEM, and CONNECT Command Rule Enhancements

Command rules now provide more `ALTER SESSION` and `ALTER SYSTEM` functionality, and CONNECT command rule enhancements.

In previous releases, you could create command rules for the `ALTER SESSION` and `ALTER SYSTEM` statements, but the functionality that you could include was limited. Starting with this release, Oracle Database Vault provides the ability to include event settings that are commonly used by `ALTER SESSION` and `ALTER SYSTEM`. The options for these special command rules enable you to use many of the features of the `ALTER SESSION` and `ALTER SYSTEM` statements, such as the ability to use the `SET` clause for parameter settings.

In addition, Oracle has made the management of the CONNECT command rule easier to maintain. In previous releases, you could create a CONNECT command rule, but if, for example, you had to create four different command rules for four different users, then you had to create a complex rule with `OR` and `AND` keywords to account for the multiple users in one command rule. With this release, you can create a CONNECT command rule for each user.

The changed and new `DBMS_MACADM` PL/SQL procedures to manage the ALTER SESSION, ALTER SYSTEM, and CONNECT command rules are as follows:

- `DBMS_MACADM.CREATE_COMMAND_RULE` (changed)
- `DBMS_MACADM.UPDATE_COMMAND_RULE` (changed)
- `DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE` (new)
- `DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE` (new)
- `DBMS_MACADM.DELETE_CONNECT_CONNECT_RULE` (new)

- `DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE` (new)
- `DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE` (new)
- `DBMS_MACADM.DELETE_SESSION_EVENT_CMD_RULE` (new)
- `DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE` (new)
- `DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE` (new)
- `DBMS_MACADM.DELETE_SYSTEM_EVENT_CMD_RULE` (new)

Related Topics

- ALTER SESSION and ALTER SYSTEM Command Rules
  You can create different kinds of `ALTER SESSION` and `ALTER SYSTEM` command rules that provide fine-grained control for these SQL statements.
- CONNECT Command Rule
  The `DBMS_MACADM.CREATE_CONNECT_CMD_RULE` procedure creates a user-specific CONNECT command rule.
- Ability to Configure Simulation Mode Protection
  Simulation mode protects Oracle Database Vault security objects so that SQL commands are not blocked, but violations to the security controls are logged.
- Ability to Create Common Realms and Common Command Rules for Oracle Multitenant
  In a multitenant environment, you now can create common realms and common command rules within the application PDB context.

Enhancements for the Authentication_Method Default Factor

Starting with this release, the Authentication_Method default factor can be used for external and global user authentication.

This enhancement provides for global and external authentication when the Kerberos and Secure Sockets Layer (SSL) strong authentication features are used.

Related Topics

- Default Factors
  Oracle Database Vault provides a set of default factors.

Changed Default Value for SQL92_SECURITY Parameter

Starting with this release, the default value for the `SQL92_SECURITY` parameter has changed from `FALSE` to `TRUE`.

> **See Also:**
>
> *Oracle Database Reference* for more information about the `SQL92_SECURITY` parameter

Oracle Database Vault Support for Flashback Technology and ILM

You now can use Oracle Flashback Technology features and Information Lifecycle Management (ILM) features in an Oracle Database Vault-enabled database.

The Oracle Flashback Technology enhancement enables you to use Database Vault realms and command rules to control access to database objects while you are using the Oracle Flashback features. You can protect the `PURGE TABLE`, `PURGE INDEX`, `FLASHBACK TABLE`, `PURGE TABLESPACE`, `PURGE RECYCLEBIN`, `PURGE DBA_RECYCLEBIN`, `CREATE FLASHBACK ARCHIVE`, `ALTER FLASHBACK ARCHIVE`, `DROP FLASHBACK ARCHIVE` SQL statements with Database Vault command rules.

The ILM enhancement enables you to use Database Vault realms and command rules with the Automatic Data Optimization (ADO) features, including granting to users the authorization to enable an ADO administrative user to perform ILM operations on Database Vault-protected objects. This enhancement enables ILM to meet regulatory compliance requirements for data retention and protection, and to store large amounts of data at the lowest cost, using storage tiering. To manage authorizations for users to perform ILM operations, two new procedures are introduced with this release: `DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER` and `DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER`. To find information about ILM authorization grants, a new data dictionary view, `DBA_DV_MAINTENANCE_AUTH`, is provided.

See Also:

- About Realms for more information about how realms are affected by this enhancement
- About Command Rules for more information about command rules
- SQL Statements That Can Be Protected by Command Rules for a list of the Oracle Flashback Technology SQL statements that can be protected by Database Vault command rules
- Using Information Lifecycle Management with Oracle Database Vault for information about granting users authorization to perform ILM tasks in a Database Vault environment
- AUTHORIZE_MAINTENANCE_USER Procedure for information about the `DBMS_MACADM.AUTHORIZE_MAINTAINANCE_USER` procedure
- UNAUTHORIZE_MAINTENANCE_USER Procedure for information about the `DBMS_MACADM.UNAUTHORIZE_MAINTAINANCE_USER` procedure
- DBA_DV_MAINTENANCE_AUTH View for information about the `DBA_DV_MAINTENANCE_USER` data dictionary view
- *Oracle Database VLDB and Partitioning Guide* for more information about ILM
- *Oracle Database Backup and Recovery User's Guide* for more information about Oracle Flashback Recovery

Support for Rolling Upgrades for Data Guard Logical Standby Databases

Oracle Data Guard logical standby databases can perform rolling upgrades for Oracle Database Vault-enabled systems using transient logical standby and the `DBMS_ROLLING` package.

See Integrating Oracle Database Vault with Oracle Data Guard for more information.

Related Topics

- Integrating Oracle Database Vault with Oracle Data Guard
  An Oracle Database Vault-Oracle Data Guard integration requires first, the primary database configuration, then the standby database configration.

Deprecated Features

The following features have been deprecated for this release.

- Deprecated Rules and Rule Sets
  Several default rules and rule sets are no longer included in a fresh installation of Oracle Database Vault.
- Deprecated UTL_FILE_DIR Parameter
  The `UTL_FILE_DIR` parameter has been deprecated for this release.

Deprecated Rules and Rule Sets

Several default rules and rule sets are no longer included in a fresh installation of Oracle Database Vault.

The following rules are no longer included in a fresh installation of an Oracle Database 12*c* release 2 (12.2) database:

- Is _system_trig_enabled Parameters Allowed
- Is o7_dictionary_accessibility Parameters Allowed
- Is _dynamic_rls_init Parameters Allowed
- Is Alter DVSYS Allowed
- Are System Security Parameters Allowed
- Are Dump or Dest Parameters Allowed
- Are Backup Restore Parameters Allowed
- Are Database File Parameters Allowed
- Are Optimizer Parameters Allowed
- Are PL-SQL Parameters Allowed
- Are Security Parameters Allowed

The following default rule sets are not included in fresh installations of Oracle Database 12*c* release 2 (12.2):

- Allow System Parameters
- Allow Fine Grained Control of System Parameters

These rules and rule sets are replaced by new rules and rule sets that are more secure.

Related Topics

- Default Rules
  Default rules are rules that have commonly used behavior, such as checking if an action evaluates to true or false.
- Default Rule Sets
  Oracle Database Vault provides a set of default rules sets that you can customize for your needs.

Deprecated UTL_FILE_DIR Parameter

The `UTL_FILE_DIR` parameter has been deprecated for this release.

This parameter is still supported for backward compatibility. However, Oracle recommends that you use directory objects instead.

1

Introduction to Oracle Database Vault

Oracle Database Vault enables you to control administrative access to your data.

- What Is Oracle Database Vault?
  Oracle Database Vault provides controls to prevent unauthorized privileged users from accessing sensitive data and to prevent unauthorized database changes.
- What Privileges Do You Need to Use Oracle Database Vault?
  Oracle Database Vault provides database roles that enable different users to perform specific tasks, based on separation-of-duty guidelines.
- Components of Oracle Database Vault
  Oracle Database Vault has a set of components that include PL/SQL packages and other special tools.
- How Oracle Database Vault Addresses Compliance Regulations
  One of the biggest side benefits resulting from regulatory compliance has been security awareness.
- How Oracle Database Vault Protects Privileged User Accounts
  Many security breaches, both external and internal, target privileged users database accounts to steal data from databases.
- How Oracle Database Vault Allows for Flexible Security Policies
  Oracle Database Vault helps you design flexible security policies for your database.
- How Oracle Database Vault Addresses Database Consolidation Concerns
  Consolidation and cloud environments reduce cost but can expose sensitive application data to those without a true need-to-know.
- How Oracle Database Vault Works in a Multitenant Environment
  To provide increased security for consolidation, you can use Oracle Database Vault with Oracle Multitenant.

What Is Oracle Database Vault?

Oracle Database Vault provides controls to prevent unauthorized privileged users from accessing sensitive data and to prevent unauthorized database changes.

- About Oracle Database Vault
  The Oracle Database Vault security controls protect application data from unauthorized access, and comply with privacy and regulatory requirements.
- Controls for Privileged Accounts
  Privileged database accounts are one of the most commonly used pathways for gaining access to sensitive applications data in the database.
- Controls for Database Configuration
  Common audit findings are unauthorized changes to database entitlements and grants of the `DBA` role to too many users.

- Enterprise Applications Protection Policies
  Application-specific Oracle Database Vault protection policies and guidelines are available for major enterprise applications.
- Run-time Privilege Analysis for Users and Applications
  Oracle Database Vault enables you to identify the actual privileges and roles that are used at run-time.

About Oracle Database Vault

The Oracle Database Vault security controls protect application data from unauthorized access, and comply with privacy and regulatory requirements.

You can deploy controls to block privileged account access to application data and control sensitive operations inside the database using trusted path authorization. Through the analysis of privileges and roles, you can increase the security of existing applications by using least privilege best practices. Oracle Database Vault secures existing database environments transparently, eliminating costly and time consuming application changes.

Controls for Privileged Accounts

Privileged database accounts are one of the most commonly used pathways for gaining access to sensitive applications data in the database.

While their broad and unrestricted access facilitates database maintenance, the same access also creates a point of attack for gaining access to large amounts of data. 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.

Figure 1-1 Oracle Database Vault Realm Blocking DBA Access to Data

Controls for Database Configuration

Common audit findings are unauthorized changes to database entitlements and grants of the `DBA` role to too many users.

Preventing unauthorized changes to production environments is important not only for security, but also for compliance as such changes can weaken security and open doors to intruders, violating privacy and compliance regulations. Oracle Database Vault SQL command rules enable you to control operations inside the database, including commands such as `CREATE TABLE`, `TRUNCATE TABLE`, and `CREATE USER`. Various out-of-the-box factors such as IP address, authentication method, and program name help implement trusted path authorization to deter attacks leveraging stolen passwords. These controls prevent accidental configuration changes and also prevent hackers and malicious insiders from tampering with applications.

The Oracle Database Vault realms with the mandatory mode enables you to seal off access to application objects, even to those with direct object grants, including the object owner. With mandatory realms, you do not need to analyze who has access because this is clear from the list of authorized users. You can enable mandatory realms at runtime and use them in response to a cyber threat, preventing all access until the threat has been analyzed.

Enterprise Applications Protection Policies

Application-specific Oracle Database Vault protection policies and guidelines are available for major enterprise applications.

These enterprise applications include Oracle Fusion Applications, Oracle E-Business Suit, Oracle PeopleSoft, Oracle Siebel, Oracle Financial Services (i-Flex), Oracle Primavera, SAP, and Finacle from Infosys.

Run-time Privilege Analysis for Users and Applications

Oracle Database Vault enables you to identify the actual privileges and roles that are used at run-time.

The additional unused roles and privileges can then be audited or revoked by the security administrators to reduce the attack surface and implement least privilege model. Privilege analysis can also be used on administrators to help limit the roles and privileges they are granted to fulfill their responsibilities. You do not need to have Oracle Database Vault enabled to perform privilege analysis.

What Privileges Do You Need to Use Oracle Database Vault?

Oracle Database Vault provides database roles that enable different users to perform specific tasks, based on separation-of-duty guidelines.

The most commonly used roles are as follows:

- `DV_OWNER` and `DV_ADMIN` enable you to create and manage Database Vault policies.
- `DV_ACCTMGR` enables you to manage user accounts.

When you register Oracle Database Vault, the `DV_OWNER` role is granted to a user who must exist before you begin the configuration process, and the `DV_ACCTMGR` role is granted to a second, optional user, who must also exist before configuration. You can grant the Database Vault roles to other users, but ensure that these users are trusted.

During the registration process, you must create backup accounts for the DV_OWNER and DV_ACCTMGR users. As a best practice, Oracle strongly recommends that you keep and maintain these backup accounts.

Related Topics

- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.
- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the DV_OWNER and DV_ACCTMGR roles.

Components of Oracle Database Vault

Oracle Database Vault has a set of components that include PL/SQL packages and other special tools.

- Oracle Database Vault Access Control Components
  Oracle Database Vault enables you to create a set of components to manage security for your database instance.
- Oracle Enterprise Manager Cloud Control Database Vault Administrator Pages
  Oracle Database Vault is pre-installed by default and can be enabled easily.
- Oracle Database Vault DVSYS and DVF Schemas
  Oracle Database Vault database objects and public functions are stored in the DVSYS and DVF schemas, respectively.
- Oracle Database Vault PL/SQL Interfaces and Packages
  Oracle Database Vault provides PL/SQL interfaces and packages for security managers or application developers to configure access control policies.
- Oracle Database Vault Reporting and Monitoring Tools
  You can generate reports on the various activities that Oracle Database Vault monitors.

Oracle Database Vault Access Control Components

Oracle Database Vault enables you to create a set of components to manage security for your database instance.

These components are as follows:

- **Realms.** A realm is a protection zone inside the database where database schemas, objects, and roles can be secured. For example, you can secure a set of schemas, objects, and roles that are related to accounting, sales, or human resources. After you have secured these into a realm, you can use the realm to control the use of system and object privileges to specific accounts or roles. This enables you to provide fine-grained access controls for anyone who wants to use these schemas, objects, and roles. Configuring Realms, discusses realms in detail. See also Oracle Database Vault Realm APIs.
- **Command rules.** A command rule is a special security policy that you can create to control how users can execute almost any SQL statement, including SELECT, ALTER SYSTEM, database definition language (DDL), and data manipulation language (DML) statements. Command rules must work with rule sets to

determine whether the statement is allowed. Configuring Command Rules , discusses command rules in detail. See also Oracle Database Vault Command Rule APIs.

- **Factors.** A factor is a named variable or attribute, such as a user location, database IP address, or session user, which Oracle Database Vault can recognize and use as a trusted path. You can use factors in rules to control activities such as authorizing database accounts to connect to the database or the execution of a specific database command to restrict the visibility and manageability of data. Each factor can have one or more identities. An identity is the actual value of a factor. A factor can have several identities depending on the factor retrieval method or its identity mapping logic. Configuring Factors, discusses factors in detail. See also Oracle Database Vault Factor APIs.
- **Rule sets.** A rule set is a collection of one or more rules that you can associate with a realm authorization, command rule, factor assignment, or secure application role. The rule set evaluates to true or false based on the evaluation of each rule it contains and the evaluation type (All True or Any True). The rule within a rule set is a PL/SQL expression that evaluates to true or false. You can have the same rule in multiple rule sets. Configuring Rule Sets , discusses rule sets in detail. See also Oracle Database Vault Rule Set APIs.
- **Secure application roles.** A secure application role is a special Oracle Database role that can be enabled based on the evaluation of an Oracle Database Vault rule set. Configuring Secure Application Roles for Oracle Database Vault, discusses secure application roles in detail. See also Oracle Database Vault Secure Application Role APIs .

To augment these components, Oracle Database Vault provides a set of PL/SQL interfaces and packages. Oracle Database Vault PL/SQL Interfaces and Packages provides an overview.

In addition to these components, you can analyze the privilege use of your users. Performing Privilege Analysis to Find Privilege Use describes how to use privilege analysis.

In general, the first step you take is to create a realm composed of the database schemas or database objects that you want to secure. You can further secure the realm by creating rules, command rules, factors, identities, rule sets, and secure application roles. In addition, you can run reports on the activities these components monitor and protect. Getting Started with Oracle Database Vault, provides a simple tutorial that will familiarize you with basic Oracle Database Vault functionality. Later chapters provide more advanced tutorials. Oracle Database Vault Reports, provides more information about how you can run reports to check the configuration and other activities that Oracle Database Vault performs.

Oracle Enterprise Manager Cloud Control Database Vault Administrator Pages

Oracle Database Vault is pre-installed by default and can be enabled easily.

Oracle Database Vault administration is fully integrated with Oracle Enterprise Manager Cloud Control, providing security administrators with a streamlined and centralized interface to manage Oracle Database Vault.

In Oracle Enterprise Manager Cloud Control, you can access the Oracle Database Vault Administrator pages if you prefer to use a graphical user interface to configure

Database Vault policies, and view Database Vault alerts and reports. Oracle Database Vault Administrator provides an extensive collection of security-related reports that assist in understanding the baseline security configuration. These reports also help point out deviations from this baseline.

Getting Started with Oracle Database Vault through DBA Operations in an Oracle Database Vault Environment explain how to use the Oracle Database Vault Administrator pages to configure access control policy defined in realms, command rules, factors, rule sets, secure application roles, and how to integrate Oracle Database Vault with other Oracle products. Monitoring Oracle Database Vault explains how to use these pages to monitor Database Vault activity, and Oracle Database Vault Reports, explains Oracle Database Vault reporting.

Oracle Database Vault DVSYS and DVF Schemas

Oracle Database Vault database objects and public functions are stored in the `DVSYS` and `DVF` schemas, respectively.

Oracle Database Vault provides a schema, `DVSYS`, which stores the database objects needed to process Oracle data for Oracle Database Vault. This schema contains the roles, views, accounts, functions, and other database objects that Oracle Database Vault uses. The `DVF` schema contains public functions to retrieve (at run time) the factor values set in the Oracle Database Vault access control configuration. Both of these schemas are authenticated as schema only accounts.

Related Topics

- Oracle Database Vault Schemas, Roles, and Accounts
  Oracle Database Vault provides schemas that contain Database Vault objects, roles that provide separation of duty for specific tasks, and default user accounts.

Oracle Database Vault PL/SQL Interfaces and Packages

Oracle Database Vault provides PL/SQL interfaces and packages for security managers or application developers to configure access control policies.

The PL/SQL procedures and functions allow the general database account to operate within the boundaries of access control policy in the context of a given database session.

See Oracle Database Vault Realm APIs through Oracle Database Vault API Reference for more information.

Oracle Database Vault Reporting and Monitoring Tools

You can generate reports on the various activities that Oracle Database Vault monitors.

In addition, you can monitor policy changes, security violation attempts, and database configuration and structural changes.

Related Topics

- Oracle Database Vault Reports
  Oracle Database Vault provides reports that track activities, such as the Database Vault configuration settings.

- Monitoring Oracle Database Vault
  You can monitor Oracle Database Vault by checking for violations to the Database Vault configurations and by tracking changes to policies.

How Oracle Database Vault Addresses Compliance Regulations

One of the biggest side benefits resulting from regulatory compliance has been security awareness.

Historically, the focus of the information technology (IT) department has been on high availability and performance. The focus on regulatory compliance has required everyone to take a step back and look at their IT infrastructure, databases, and applications from a security angle. Common questions include:

- Where is the sensitive information stored?
- Who has access to this information?

Regulations such as the Sarbanes-Oxley Act, Health Insurance Portability and Accountability Act (HIPAA), International Convergence of Capital Measurement and Capital Standards: a Revised Framework (Basel II), Japan Privacy Law, Payment Card Industry Data Security Standard (PCI DSS), and the European Union Directive on Privacy and Electronic Communications have common themes that include internal controls, separation of duty, and access control.

While most changes required by regulations such as Sarbanes-Oxley and HIPAA are procedural in nature, the remainder may require technology investments. A common security requirement found in regulations is stringent internal controls. The degree to which Oracle Database Vault helps an organization achieve compliance varies with the regulation. In general, Oracle Database Vault realms, command rules, factors and separation of duty features, help reduce the overall security risks that regulation provisions worldwide address.

Table 1-1 lists regulations that address potential security threats.

Table 1-1 Regulations That Address Potential Security Threats

Regulation	Potential Security Threat
Sarbanes-Oxley Section 302	Unauthorized changes to data
Sarbanes-Oxley Section 404	Modification to data, unauthorized access
Sarbanes-Oxley Section 409	Denial of service, unauthorized access
Gramm-Leach-Bliley	Unauthorized access, modification, or disclosure
Health Insurance Portability and Accountability Act (HIPAA) 164.306	Unauthorized access to data
HIPAA 164.312	Unauthorized access to data
Basel II – Internal Risk Management	Unauthorized access to data
CFR Part 11	Unauthorized access to data
Japan Privacy Law	Unauthorized access to data
EU Directive on Privacy and Electronic Communications	Unauthorized access to data

Table 1-1 (Cont.) Regulations That Address Potential Security Threats

Regulation	Potential Security Threat
Payment Card Industry Data Security Standard (PCI DSS)	Unauthorized changes to data

How Oracle Database Vault Protects Privileged User Accounts

Many security breaches, both external and internal, target privileged users database accounts to steal data from databases.

Oracle Database Vault protects against compromised privilege user account attacks by using realms, factors, and command rules. Combined, these provide powerful security tools to help secure access to databases, applications, and sensitive information. You can combine rules and factors to control the conditions under which commands in the database are allowed to execute, and to control access to data protected by a realm. For example, you can create rules and factors to control access to data based on IP addresses, the time of day, and specific programs. These can limit access to only those connections that pass these conditions. This can prevent unauthorized access to application data and access to the database by unauthorized applications.

Oracle Database Vault provides built-in factors that you can use in combination with rules to control access to the database, realm-protected applications, and commands within the database.

You can associate rules and factors with many SQL statements in the database to provide stronger internal controls within the database. You can customize these to meet the operational policies for your site. For example, you could define a rule to limit execution of the `ALTER SYSTEM` statement to a specific IP address and host name.

How Oracle Database Vault Allows for Flexible Security Policies

Oracle Database Vault helps you design flexible security policies for your database.

For example, any database user who has the `DBA` role can make modifications to basic parameters in a database. Suppose an inexperienced administrator who has system privileges decides to start a new redo log file but does not realize that doing so at a particular time may cause problems for the database. With Oracle Database Vault, you can create a command rule to prevent this user from making such modifications by limiting his or her usage of the `ALTER SYSTEM SWITCH LOGFILE` statement. Furthermore, you can attach rule sets to the command rule to restrict activity further, such as limiting the statement's execution in the following ways:

- By time (for example, only during 4 p.m. and 5 p.m. on Friday afternoons)
- By local access only, that is, not remotely
- By IP address (for example, allowing the action to only a specified range of IP addresses)

You can customize Oracle Database Vault separation of duties to fit the requirements of business of any size. For example, large customers with dedicated IT staff and some out sourced back end operations can further fine tune separation of duties to control what out sourced database administrators can do. For smaller organizations with some users handling multiple responsibilities, separation of duties can be tuned down and these users can create separate dedicated accounts for each responsibility. This helps such users keep track of all actions made and prevents intruders from exploiting compromised privileged database accounts to steal sensitive data. In addition, it helps auditors verify compliance.

How Oracle Database Vault Addresses Database Consolidation Concerns

Consolidation and cloud environments reduce cost but can expose sensitive application data to those without a true need-to-know.

Data from one country may be hosted in an entirely different country, but access to that data must be restricted based on regulations of the country to which the data belongs. Oracle Database Vault controls provide increased security for these environments by preventing database administrators from accessing the applications data. In addition, controls can be used to help block application bypass and enforce a trusted-path from the application tier to the application data.

Oracle Database Vault provides four distinct separation of duty controls for security administration:

- Day-to-day database administrator tasks using the default Oracle Database `DBA` role
- Security administrator tasks using the `DV_OWNER` and `DV_ADMIN` roles
- Account administrator tasks using the `DV_ACCTMGR` role
- Grants of roles and privileges by a named trusted user

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.

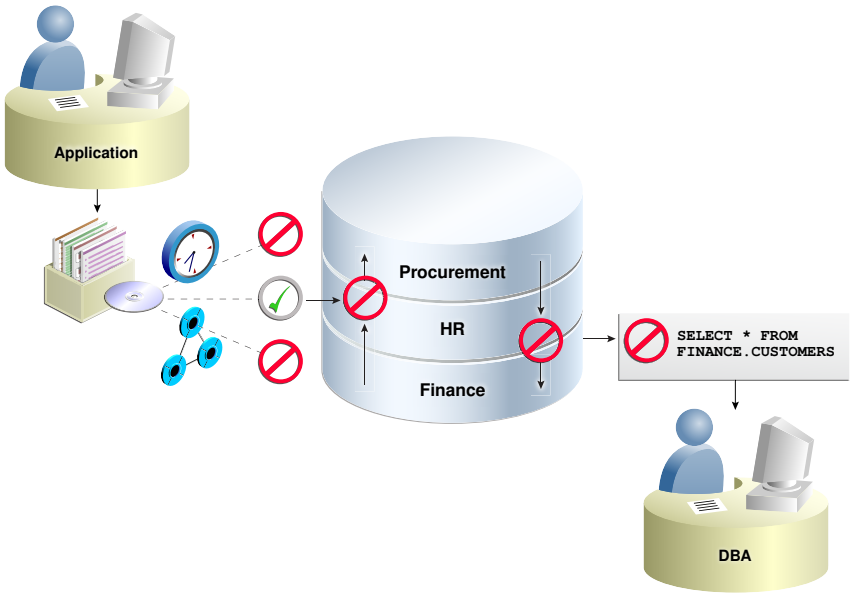
Oracle customers today still have hundreds and even thousands of databases distributed throughout the enterprise and around the world. However, for database consolidation as a cost-saving strategy in the coming years to be effective, the physical security provided by the distributed database architecture must be available in the consolidated environment. Oracle Database Vault addresses the primary security concerns of database consolidation.

Figure 1-2 illustrates how Oracle Database Vault addresses the following database security concerns:

- **Administrative privileged account access to application data**: In this case, Oracle Database Vault prevents the database administrator from accessing the schemas that are protected by the Finance realm. Although the database administrator is the most powerful and trusted user, this administrator does not need access to application data residing within the database.

- **Separation of duties for application data access**: In this case, the HR realm owner, created in Oracle Database Vault, has access to the HR realm schemas.

Figure 1-2 Oracle Database Vault Security



Database consolidation can result in multiple powerful user accounts residing in a single database. This means that in addition to the overall database administrator, individual application schema owners also may have powerful privileges. Revoking some privileges may adversely affect existing applications. Using Oracle Database Vault realms, you can enforce access to applications through a trusted path, preventing database users who have not been specifically authorized access from using powerful privileges to look at other application data. For example, a database administrator who has the `SELECT ANY TABLE` system privilege can be prevented from using that privilege to view other application data residing in the same database.

How Oracle Database Vault Works in a Multitenant Environment

To provide increased security for consolidation, you can use Oracle Database Vault with Oracle Multitenant.

Oracle Database Vault can prevent privileged user access inside a pluggable database (PDB) and between the PDB and the common privileged user at the container database. Each PDB has its own Database Vault metadata, such as realms, rule sets, command rules, default policies (such as default realms), and so on. In addition, the objects within the `DVSYS` and `DVF` schemas are automatically available to any child PDBs. Both schemas are common user schemas.

You can configure common realms in the application root only, but you can create common rule sets and command rules in either the application root or the CDB root. A common command rule in the application root applies to its associated PDBs, and common command rules in the CDB root apply to all PDBs in the CDB environment.

The ability to create common realms and command rules enables you to create policies that use a shared set of realms, rule sets, or command rules throughout the CDB environments, rather than having to create these same components for every PDB in the multitenant environment. The common protection applies for all PDBs associated with the application root that have Oracle Database Vault enabled.

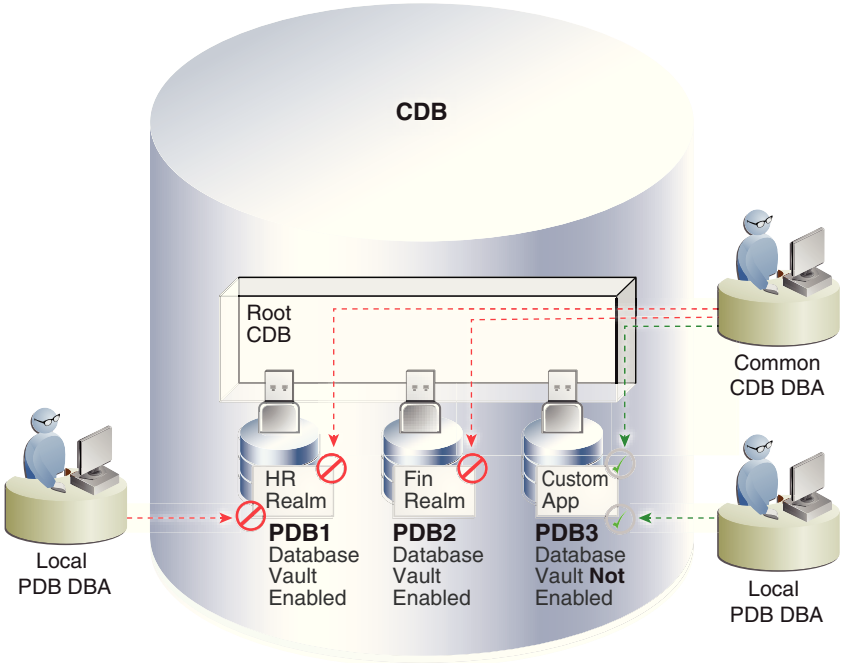
You can create individual local policies for each PDB. When you use Database Vault to protect an object, Database Vault subjects common privileges for common objects to the same enforcement rules as local system privileges.

When you configure a PDB that has Database Vault enabled, the `DVSYS` schema is a common user schema that is stored in the root. This means that all the objects within the `DVSYS` schema (tables, data dictionary views, user accounts, PL/SQL packages, default policies, and so on) are subject to the common privileges available for this schema. In other words, you can create realms, factors, and so on in the root to protect the schema in the root. Ensure that you configure Database Vault in the root first, before you configure it in the associated PDBs.

When you enable Oracle Database Vault in the CDB root, you can choose either regular mode or strict mode. The settings propagate throughout the CDB based on the setting you choose. For example, suppose a CDB contains both Database Vault-enabled PDBs and PDBs in which Database Vault is not enabled. If you enable Database Vault using regular mode, then both types of PDBs continue to function normally. If you enable Database Vault using strict mode, then the Database Vault-disabled PDBs operate in restricted mode.

Figure 1-3 illustrates how the database in regular mode allows different access for common and local database administrators depending if Database Vault is enabled. In this scenario, neither the common user nor the local users have access to the realms in PDB1 and PDB2. Both the common user and the PDB3 local user have access to the Custom App application in PDB3, where Database Vault is not enabled.

Figure 1-3 Oracle Database Vault in a Multitenant Environment with Regular Mode



Related Topics

- Realms in a Multitenant Environment
  In a multitenant environment, you can create a realm to protect common objects in the application root.
- Rule Sets and Rules in a Multitenant Environment
  In a multitenant environment, you can create a rule set and its associated rules in the application root.
- Command Rules in a Multitenant Environment
  In a multitenant environment, you can create common and local command rules in either the CDB root or the application root.
- Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB
  You can convert a standalone Oracle Database Release 12c or later database to a PDB, and then plug this PDB into a CDB.

2

What to Expect After You Enable Oracle Database Vault

When you enable Oracle Database Vault, several Oracle Database security features, such as default user authorizations, are modified to provide stronger security restrictions.

- Initialization and Password Parameter Settings That Change
  The Oracle Database Vault configuration modifies several database initialization parameter settings to better secure your database configuration.
- How Oracle Database Vault Restricts User Authorizations
  The Oracle Database configuration requires two additional administrative database account names.
- New Database Roles to Enforce Separation of Duties
  The Oracle Database Vault configuration implements the concept of *separation of duty* so that you can meet regulatory, privacy and other compliance requirements.
- Privileges That Are Revoked from Existing Users and Roles
  The Oracle Database Vault configuration revokes privileges from several Oracle Database-supplied users and roles, for better separation of duty.
- Privileges That Are Prevented for Existing Users and Roles
  The Oracle Database Vault configuration prevents several privileges for all users and roles who have been granted these privileges, including users `SYS` and `SYSTEM`.
- Modified AUDIT Statement Settings for a Non-Unified Audit Environment
  When you configure Oracle Database Vault and if you decide not to use unified auditing, then Database Vault configures several `AUDIT` statements.

Initialization and Password Parameter Settings That Change

The Oracle Database Vault configuration modifies several database initialization parameter settings to better secure your database configuration.

If these changes adversely affect your organizational processes or database maintenance procedures, then contact Oracle Support for help in resolving the issue.

Table 2-1 describes the initialization parameter settings that Oracle Database Vault modifies. Initialization parameters are stored in the `init.ora` initialization parameter file. On UNIX and Linux, this file is located in `$ORACLE_HOME/dbs`. On Windows, this file is located in `$ORACLE_HOME/database`. For more information about this file, see *Oracle Database Reference*.

Table 2-1 Modified Database Initialization Parameter Settings

Parameter	Default Value in Database	New Value Set by Database Vault	Impact of the Change
AUDIT_SYS_OPERATIONS	FALSE	TRUE	Enables the auditing of top-level operations directly issued by user SYS, and users connecting with SYSDBA or SYSOPER privilege. For more information about AUDIT_SYS_OPERATIONS, see *Oracle Database Reference*.
OS_ROLES	Not configured	FALSE	Disables the operating system to completely manage the granting and revoking of roles to users. Any previous grants of roles to users using GRANT statements do not change, because they are still listed in the data dictionary. Only the role grants made at the operating system-level to users apply. Users can still grant privileges to roles and users. For more information about OS_ROLES, see *Oracle Database Reference*.
REMOTE_LOGIN_PASSWORDFILE	EXCLUSIVE	EXCLUSIVE	Specifies whether Oracle Database checks for a password file. The EXCLUSIVE setting enforces the use of the password file, if you installed Oracle Database Vault into a database where REMOTE_LOGIN_PASSWORDFILE is not set to EXCLUSIVE. For more information about REMOTE_LOGIN_PASSWORDFILE, see *Oracle Database Reference*.
SQL92_SECURITY	TRUE	TRUE	Ensures that if a user has been granted the UPDATE or DELETE object privilege, then the user must also be granted the SELECT object privilege before being able to perform UPDATE or DELETE operations on tables that have WHERE or SET clauses. Be aware that if the user is only granted the READ object privilege (instead of SELECT), then the user is not able to perform UPDATE or DELETE operations. For more information about SQL92_SECURITY, see *Oracle Database Reference*.

How Oracle Database Vault Restricts User Authorizations

The Oracle Database configuration requires two additional administrative database account names.

In addition, several database roles are created. These accounts are part of the separation of duties provided by Oracle Database Vault. One common audit problem that has affected several large organizations is the unauthorized creation of new database accounts by a database administrator within a production instance. Upon installation, Oracle Database Vault prevents anyone other than the Oracle Database Vault account manager or a user granted the Oracle Database Vault account manager role from creating users in the database.

Related Topics

- Separation of Duty Guidelines
  Oracle Database Vault is designed to easily implement separation of duty guidelines.

New Database Roles to Enforce Separation of Duties

The Oracle Database Vault configuration implements the concept of *separation of duty* so that you can meet regulatory, privacy and other compliance requirements.

Oracle Database Vault makes clear separation between the account management responsibility, data security responsibility, and database management responsibility inside the database. This means that the concept of a super-privileged user (for example, `DBA`) is divided among several new database roles to ensure no one user has full control over both the data and configuration of the system. Oracle Database Vault prevents privileged users (those with the `DBA` and other privileged roles and system privileges) from accessing designated protected areas of the database called realms. It also introduces new database roles called the Oracle Database Vault Owner (`DV_OWNER`) and the Oracle Database Vault Account Manager (`DV_ACCTMGR`). These new database roles separate the data security and the account management from the traditional `DBA` role. You should map these roles to distinct security professionals within your organization.

Related Topics

- Separation of Duty Guidelines
  Oracle Database Vault is designed to easily implement separation of duty guidelines.
- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.
- Oracle Database Vault Accounts Created During Registration
  You must create accounts for the Oracle Database Vault Owner and Oracle Database Vault Account Manager during the registration process.

Privileges That Are Revoked from Existing Users and Roles

The Oracle Database Vault configuration revokes privileges from several Oracle Database-supplied users and roles, for better separation of duty.

Table 2-2 lists privileges that Oracle Database Vault revokes from the Oracle Database-supplied users and roles. Be aware that if you disable Oracle Database Vault, these privileges remain revoked. If your applications depend on these privileges, then grant them to application owner directly. In a multitenant environment, these privileges are revoked from the users and roles in the CDB root and its PDBs and from the application root and its PDBs.

Table 2-2 Privileges Oracle Database Vault Revokes

User or Role	Privilege That Is Revoked
DBA role	• BECOME USER • SELECT ANY TRANSACTION • CREATE ANY JOB • CREATE EXTERNAL JOB • EXECUTE ANY PROGRAM • EXECUTE ANY CLASS • MANAGE SCHEDULER • DEQUEUE ANY QUEUE • ENQUEUE ANY QUEUE • MANAGE ANY QUEUE
IMP_FULL_DATABASE role[1]	• BECOME USER • MANAGE ANY QUEUE
EXECUTE_CATALOG_ROLE role	• EXECUTE ON DBMS_LOGMNR • EXECUTE ON DBMS_LOGMNR_D • EXECUTE ON DBMS_LOGMNR_LOGREP_DICT • EXECUTE ON DBMS_LOGMNR_SESSION • EXECUTE ON DBMS_FILE_TRANSFER
PUBLIC user	• EXECUTE ON UTL_FILE
SCHEDULER_ADMIN role[2]	• CREATE ANY JOB • CREATE EXTERNAL JOB • EXECUTE ANY PROGRAM • EXECUTE ANY CLASS • MANAGE SCHEDULER

1 To authorize users to export and import data using Oracle Data Pump, see Using Oracle Data Pump with Oracle Database Vault.
2 To authorize users to schedule database jobs, see Using Oracle Scheduler with Oracle Database Vault.

Note:

Both the SYS and SYSTEM users retain the SELECT privilege for the DBA_USERS_WITH_DEFPWD data dictionary view, which lists user accounts that use default passwords. If you want other users to have access to this view, grant them the SELECT privilege on it.

Related Topics

- Table 14-1

- DV_ACCTMGR Database Vault Account Manager Role
  The DV_ACCTMGR role is a powerful role, used for accounts management.

Privileges That Are Prevented for Existing Users and Roles

The Oracle Database Vault configuration prevents several privileges for all users and roles who have been granted these privileges, including users SYS and SYSTEM.

The DV_ACCTMGR role has these privileges for separation of duty.

- ALTER PROFILE
- ALTER USER
- CREATE PROFILE
- CREATE USER
- DROP PROFILE
- DROP USER

For better security and to maintain separation-of-duty standards, do not enable SYS or SYSTEM users the ability to create or manage user accounts.

Modified AUDIT Statement Settings for a Non-Unified Audit Environment

When you configure Oracle Database Vault and if you decide not to use unified auditing, then Database Vault configures several AUDIT statements.

Related Topics

- Oracle Database Audit Settings Created for Oracle Database Vault
  When you install Oracle Database Vault, it creates several AUDIT settings in the database.

3
Getting Started with Oracle Database Vault

Before you can start using Oracle Database Vault, you must register it with the Oracle database.

- About Registering Oracle Database Vault with an Oracle Database
  After you install Oracle Database, you must register (that is, configure and enable) Oracle Database Vault with the Oracle database in which it was installed.
- Registering Oracle Database Vault with an Oracle Database in a Multitenant Environment
  You can register Oracle Database Vault in a multitenant environment based on several scenarios.
- Registering Database Vault in a Non-Multitenant Environment
  You can register Oracle Database Vault from SQL*Plus in a non-multitenant environment.
- Verifying That Database Vault Is Configured and Enabled
  The `DBA_DV_STATUS`, `CDB_DV_STATUS`, `DBA_OLS_STATUS`, and `CDB_OLS_STATUS` data dictionary views verify if Oracle Database is configured and enabled.
- Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control
  Oracle Enterprise Manager Cloud Control (Cloud Control) provides pages for managing Oracle Database Vault.
- Quick Start Tutorial: Securing a Schema from DBA Access
  This tutorial shows how to create a realm around the `HR` schema.

About Registering Oracle Database Vault with an Oracle Database

After you install Oracle Database, you must register (that is, configure and enable) Oracle Database Vault with the Oracle database in which it was installed.

Oracle Database includes Database Vault when you choose to include a default database in the installation process, but you must register it before you can use it. If you create a custom database, then you can use DBCA to install and enable Database Vault for it. The registration process enables Oracle Label Security if it is not already enabled. Oracle Label Security is required for Oracle Database Vault but it does not require a separate license unless you begin using Oracle Label Security separately and create Oracle Label Security policies. This procedure applies to the CDB root, application root, and the current pluggable database (PDB), as well as to both single-instance and Oracle Real Application Clusters (Oracle RAC) installations. In a multitenant database, Database Vault must be configured with the CDB root before any of the PDBs can configure Database Vault.

As part of the registration process, you created the Database Vault backup accounts. These are accounts that hold the key Database Vault roles. Use these accounts initially to provision the roles to named users with administrative privileges. Maintaining a backup account will allow you to recover from the named user losing or somehow

misplacing their credentials because SYS will not be able to reset these passwords for users with these roles.

You can register Oracle Database in both multitenant and non-multitenant environments. For multitenant environments, you have several methods to choose from for the registration.

> **Note:**
>
> If you have upgraded from a release earlier than Oracle Database 12*c*, and if the earlier Oracle Database Vault had been enabled in that earlier release, then after the upgrade process is complete, you must enable Oracle Database Vault by using the DBMS_MACADM.ENABLE_DV procedure.
> In a multitenant environment, if you are migrating a non-Database Vault registered Oracle database from a release earlier than release 12c, then you must perform a manual installation of Database Vault.

Related Topics

- Verifying That Database Vault Is Configured and Enabled
  The DBA_DV_STATUS, CDB_DV_STATUS, DBA_OLS_STATUS, and CDB_OLS_STATUS data dictionary views verify if Oracle Database is configured and enabled.

Registering Oracle Database Vault with an Oracle Database in a Multitenant Environment

You can register Oracle Database Vault in a multitenant environment based on several scenarios.

- About Registering Database Vault in a Multitenant Environment
  In a multitenant environment, you must register Oracle Database Vault in the CDB root before you can register Database Vault in any of the associated PDBs.
- Registering Database Vault with Common Users to Manage the CDB Root
  In a multitenant environment, you can register Oracle Database Vault with a common user who will manage the CDB root.
- Registering Database Vault Common Users to Manage Specific PDBs
  In a multitenant environment, you must register Oracle Database Vault in the root first, then in the PDBs afterward.
- Plugging in a Database Vault-Enabled PDB
  From SQL*Plus, in a multitenant environment, you can plug in a database that already has Database Vault enabled.
- Manually Installing Oracle Database Vault in a Multitenant Environment
  Under certain conditions, for a multitenant environment, you must manually install Oracle Database Vault.

About Registering Database Vault in a Multitenant Environment

In a multitenant environment, you must register Oracle Database Vault in the CDB root before you can register Database Vault in any of the associated PDBs.

The common users who have been assigned the DV_OWNER and DV_ACCTMGR roles in the CDB root can also have the same role in the PDBs. PDBs can have Database Vault registered using the same common users or use separate PDB local users. The DV_ACCTMGR role is granted commonly to the common user in the CDB root. You can grant DV_OWNER locally or commonly to the CDB root common user when you register Database Vault with the CDB root. Granting DV_OWNER locally to the common user prevents the common DV_OWNER user from using this role in any PDB.

Registering Database Vault with Common Users to Manage the CDB Root

In a multitenant environment, you can register Oracle Database Vault with a common user who will manage the CDB root.

1. In a multitenant environment, log into the root of the database instance as a user who has privileges to create users and grant the CREATE SESSION and SET CONTAINER privileges.

   For example:

   ```
   sqlplus c##sec_admin
   Enter password: password
   ```

2. Create accounts that will be used for the Database Vault Owner (DV_OWNER role) and Database Vault Account Manager (DV_ACCTMGR role) accounts.

   Oracle strongly recommends that you create two accounts for each role. One account, the primary account, will be used on a day-to-day basis and the other account will be used as a backup account in case the password of the primary account is lost and must be reset.

   Prepend the names of these accounts with c## or C##. For example:

   ```
   GRANT CREATE SESSION, SET CONTAINER TO c##dbv_owner_root IDENTIFIED BY password
   CONTAINER = ALL;
   GRANT CREATE SESSION, SET CONTAINER TO c##dbv_owner_root_backup IDENTIFIED BY
   password CONTAINER = ALL;
   GRANT CREATE SESSION, SET CONTAINER TO c##dbv_acctmgr_root IDENTIFIED BY
   password CONTAINER = ALL;
   GRANT CREATE SESSION, SET CONTAINER TO c##dbv_acctmgr_root_backup IDENTIFIED BY
   password CONTAINER = ALL;
   ```

   Replace *password* with a password that is secure. See *Oracle Database Security Guide* for the minimum requirements for creating passwords.

3. Connect to the root as user SYS with the SYSDBA administrative privilege

   ```
   CONNECT SYS AS SYSDBA
   Enter password: password
   ```

4. Configure the primary Database Vault user accounts.

   ```
   BEGIN
    CONFIGURE_DV (
      dvowner_uname         => 'c##dbv_owner_root',
      dvacctmgr_uname       => 'c##dbv_acctmgr_root');
    END;
   /
   ```

5. Run the utlrp.sql script to recompile invalidated objects in the root.

```
@?/rdbms/admin/utlrp.sql
```

If the script provides instructions, follow them, and then run the script again. If the script terminates abnormally without giving any instructions, run it again.

6. Connect to the root as the primary Database Vault Owner user that you just configured.

   For example:

   ```
   CONNECT c##dbv_owner_root_backup
   Enter password: password
   ```

7. Enable Oracle Database Vault using one of the following options:

   - To enable Oracle Database Vault to use regular mode, which puts the PDB in restricted mode until you enable Database Vault in the PDB and then restart the PDB:

     ```
     EXEC DBMS_MACADM.ENABLE_DV (strict_mode => 'n');
     ```

   - To enable Oracle Database Vault to use strict mode, which enables Database Vault in each PDB:

     ```
     EXEC DBMS_MACADM.ENABLE_DV (strict_mode => 'y');
     ```

8. Connect with the SYSDBA administrative privilege.

   ```
   CONNECT / AS SYSDBA
   ```

9. Restart the database.

   ```
   SHUTDOWN IMMEDIATE
   STARTUP
   ```

10. Connect as the primary DV_OWNER user and then grant the DV_OWNER role to the backup DV_OWNER user that you created earlier.

    For example:

    ```
    CONNECT c##dbv_owner_root
    Enter password: password

    GRANT DV_OWNER TO c##dbv_owner_root_backup WITH ADMIN OPTION;
    ```

11. Connect as the primary DV_ACCTMGR user and then grant the DV_ACCTMGR role to the backup DV_ACCTMGR user.

    For example:

    ```
    CONNECT c##dbv_acctmgr_root
    Enter password: password

    GRANT DV_ACCTMGR TO c##dbv_acctmgr_root_backup WITH ADMIN OPTION;
    ```

12. Store the two backup account passwords in a safe location in case they are needed in the future.

Related Topics

- Verifying That Database Vault Is Configured and Enabled
  The DBA_DV_STATUS, CDB_DV_STATUS, DBA_OLS_STATUS, and CDB_OLS_STATUS data dictionary views verify if Oracle Database is configured and enabled.
- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.

- Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control
  Oracle Enterprise Manager Cloud Control (Cloud Control) provides pages for managing Oracle Database Vault.

Registering Database Vault Common Users to Manage Specific PDBs

In a multitenant environment, you must register Oracle Database Vault in the root first, then in the PDBs afterward.

If you try to register in a PDB first, then an `ORA-47503: Database Vault is not enabled on CDB$ROOT` error appears.

1. In a multitenant environment, log into the root of the database instance as a user who has privileges to create users and to grant the `CREATE SESSION` and `SET CONTAINER` privileges.

   For example:

   ```
   sqlplus c##sec_admin
   Enter password: password
   ```

2. If you have not already done so, then create user accounts to be used as the Database Vault accounts.

   See Step 2 under Registering Database Vault with Common Users to Manage the CDB Root for more information about creating these accounts.

3. Ensure that you have registered Oracle Database Vault in the CDB root, as described in Registering Database Vault with Common Users to Manage the CDB Root.

4. Connect to the PDB to which the common users will need access.

   For example:

   ```
   CONNECT c##sec_admin@pdb_name
   Enter password: password
   ```

   To find the available PDBs, query the `DBA_PDBS` data dictionary view. To check the current PDB, run the `show con_name` command.

5. Grant the `CREATE SESSION` and `SET CONTAINER` privileges to the users for this PDB.

   For example:

   ```
   GRANT CREATE SESSION, SET CONTAINER TO c##dbv_owner_root CONTAINER = CURRENT;
   GRANT CREATE SESSION, SET CONTAINER TO c##dbv_acctmgr_root CONTAINER = CURRENT;
   ```

6. Connect as user `SYS` with the `SYSDBA` administrative privilege

   ```
   CONNECT SYS@pdb_name AS SYSDBA
   Enter password: password
   ```

7. While still in the PDB, configure the two primary Database Vault user accounts.

   For example:

   ```
   BEGIN
    CONFIGURE_DV (
      dvowner_uname         => 'c##dbv_owner_root',
      dvacctmgr_uname       => 'c##dbv_acctmgr_root');
    END;
   /
   ```

8. Run the `utlrp.sql` script to recompile invalidated objects in this PDB.

```
@?/rdbms/admin/utlrp.sql
```

If the script provides instructions, follow them, and then run the script again. If the script terminates abnormally without giving any instructions, run it again.

9. Connect to the PDB as the primary Database Vault Owner user that you just configured.

   For example:

   ```
   CONNECT c##dbv_owner_root@pdb_name
   Enter password: password
   ```

10. Enable Oracle Database Vault in this PDB.

    ```
    EXEC DBMS_MACADM.ENABLE_DV;
    ```

11. Connect to the CDB with the SYSDBA administrative privilege.

    ```
    CONNECT / AS SYSDBA
    ```

12. Close and reopen the PDB.

    For example:

    ```
    ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;
    ALTER PLUGGABLE DATABASE pdb_name OPEN;
    ```

Related Topics

- Verifying That Database Vault Is Configured and Enabled
  The DBA_DV_STATUS, CDB_DV_STATUS, DBA_OLS_STATUS, and CDB_OLS_STATUS data dictionary views verify if Oracle Database is configured and enabled.
- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.
- Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control
  Oracle Enterprise Manager Cloud Control (Cloud Control) provides pages for managing Oracle Database Vault.

Plugging in a Database Vault-Enabled PDB

From SQL*Plus, in a multitenant environment, you can plug in a database that already has Database Vault enabled.

In this scenario, the plugged in database has its own local Database Vault accounts. Be aware that if you plug a Database Vault-enabled database into a CDB that is not Database Vault enabled, then the PDB will remain in restricted mode until you enable Database Vault in the CDB and then restart the CDB. If you plug a non-Database Vault-enabled PDB into a CDB that is Database Vault enabled, then the PDB remains in restricted mode until you enable Database Vault in the PDB and then restart the PDB. This plugged in non-Database Vault enabled PDB can still be used. However, if the CDB is Database Vault enabled with the strict option set, then the PDB must be Database Vault enabled.

Before you plug in a Database Vault-enabled PDB and if the Database Vault roles are granted to common users, ensure that you understand fully how plugging in PDBs affect common users.

Related Topics

- *Oracle Database Security Guide*

Manually Installing Oracle Database Vault in a Multitenant Environment

Under certain conditions, for a multitenant environment, you must manually install Oracle Database Vault.

For example, you must manually install Oracle Database Vault if a release 11g Oracle database without Database Vault is upgraded to release 12c, then converted to a PDB to be plugged into a 12c Database Vault-enabled database. In addition, you must manually install Oracle Database Vault (and Oracle Label Security) in a PDB if this PDB does not have these products when the PDB has been plugged into a CDB where Database Vault and Label Security are installed.

1. As user who has been granted the `SYSDBA` administrative privilege, log in to the PDB in which you want to install Oracle Database Vault.

   For example, to log in to a PDB named `hr_pdb`:

   ```
   sqlplus sec_admin@hr_pdb as sysdba
   Enter password: password
   ```

   To find the available PDBs, run the `show pdbs` command. To check the current PDB, run the `show con_name` command.

2. If necessary, check if Oracle Database Vault and Oracle Label Security are already installed on this PDB.

   If the `DVSYS` account (for Database Vault) and the `LBACSYS` account (for Label Security) exist, then Database Vault and Label Security exist on the PDB.

   ```
   SELECT USERNAME FROM DBA_USERS WHERE USERNAME IN ('DVSYS', 'LBACSYS');
   ```

3. If neither Database Vault nor Label Security have been installed, then install Oracle Label Security by executing the `catols.sql` script.

   ```
   @$ORACLE_HOME/rdbms/admin/catols.sql
   ```

   Oracle Label Security must be installed before you can install Oracle Database Vault.

4. Install Oracle Database Vault by executing the `catmac.sql` script.

   ```
   @$ORACLE_HOME/rdbms/admin/catmac.sql
   ```

5. At the `Enter value for 1` prompt, enter `SYSTEM` as the tablespace to install `DVSYS`.
6. At the `Enter value for 2` prompt, enter the temporary tablespace for the PDB.

After the installation is complete, you can register Oracle Database Vault in the PDB. If Database Vault is not registered in the CDB already, you must close the PDB before you can register Database Vault in the CDB root. Database Vault must be registered in CDB root before it can be registered in the PDB. After Database Vault is registered in

the CDB root and the database has been restarted, then you can open the PDB and register Database Vault.

Related Topics

- Registering Oracle Database Vault with an Oracle Database in a Multitenant Environment
  You can register Oracle Database Vault in a multitenant environment based on several scenarios.

Registering Database Vault in a Non-Multitenant Environment

You can register Oracle Database Vault from SQL*Plus in a non-multitenant environment.

1. Log into the database instance as a user who has privileges to create user accounts and grant the `CREATE SESSION` privilege to other users.

   For example:

   ```
   sqlplus sec_admin
   Enter password: password
   ```

2. Identify (or create new named user accounts if necessary) named user accounts to be used for the Database Vault Owner (`DV_OWNER` role) and Database Vault Account Manager (`DV_ACCTMGR` role) accounts.

   Oracle strongly recommends that you create two accounts for each role. One account, the primary named user account, will be used on a day-to-day basis and the other account will be used as a backup account in case the password of the primary account is lost and must be reset.

   For example:

   ```
   GRANT CREATE SESSION TO sec_admin_owen IDENTIFIED BY password;
   GRANT CREATE SESSION TO dbv_owner_backup IDENTIFIED BY password;
   GRANT CREATE SESSION TO accts_admin_ace IDENTIFIED BY password;
   GRANT CREATE SESSION TO dbv_acctmgr_backup IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

3. Connect with the `SYSDBA` administrative privilege.

   ```
   CONNECT / AS SYSDBA
   Enter password: password
   ```

4. Configure Database Vault using the two backup user accounts.

   ```
   BEGIN
    CONFIGURE_DV (
      dvowner_uname         => 'dbv_owner_backup',
      dvacctmgr_uname       => 'dbv_acctmgr_backup');
    END;
   /
   ```

   Do not enter the names `DV_OWNER`, `DV_ACCTMGR`, or the names of any other Database Vault roles for these user accounts.

5. Run the `utlrp.sql` script to recompile invalidated objects.

```
@?/rdbms/admin/utlrp.sql
```

If the script gives you any instructions, follow them, and then run the script again. If the script terminates abnormally without giving any instructions, run it again.

6. Connect as the backup Database Vault Owner user that you just configured.

   For example:

   ```
   CONNECT dbv_owner_backup
   Enter password: password
   ```

7. Enable Oracle Database Vault.

   ```
   EXEC DBMS_MACADM.ENABLE_DV;
   ```

8. Connect with the SYSDBA administrative privilege.

   ```
   CONNECT / AS SYSDBA
   ```

9. Restart the database.

   ```
   SHUTDOWN IMMEDIATE
   STARTUP
   ```

10. Connect as the backup DV_OWNER user and then grant the DV_OWNER role to the primary DV_OWNER user that you created earlier.

    For example:

    ```
    CONNECT dbv_owner_backup
    Enter password: password

    GRANT DV_OWNER TO sec_admin_owen WITH ADMIN OPTION;
    ```

11. Connect as the backup DV_ACCTMGR user and then grant the DV_ACCTMGR role to the primary DV_ACCTMGR user.

    For example:

    ```
    CONNECT dbv_acctmgr_backup
    Enter password: password

    GRANT DV_ACCTMGR TO accts_admin_ace WITH ADMIN OPTION;
    ```

12. Verify that the configuration was successful.

    ```
    CONNECT / AS SYSDBA

    SELECT * FROM DBA_DV_STATUS;
    SELECT * FROM DBA_OLS_STATUS;
    ```

13. Store the two backup account passwords in a safe location such as a privileged account management (PAM) system in case they are needed in the future.

Related Topics

- Verifying That Database Vault Is Configured and Enabled
  The DBA_DV_STATUS, CDB_DV_STATUS, DBA_OLS_STATUS, and CDB_OLS_STATUS data dictionary views verify if Oracle Database is configured and enabled.
- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.

- Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control
  Oracle Enterprise Manager Cloud Control (Cloud Control) provides pages for managing Oracle Database Vault.

Verifying That Database Vault Is Configured and Enabled

The DBA_DV_STATUS, CDB_DV_STATUS, DBA_OLS_STATUS, and CDB_OLS_STATUS data dictionary views verify if Oracle Database is configured and enabled.

In addition to Oracle Database Vault administrators, the Oracle Database SYS user and users who have been granted the DBA role can query these views.

- For Database Vault:
  - If you want to find the Database Vault status for a non-multitenant database, or in a multitenant environment for the root only or an individual PDB, then query DBA_DV_STATUS. For example:

    ```
    SELECT * FROM DBA_DV_STATUS;
    ```

    Output similar to the following appears:

    ```
    NAME                 STATUS
    -------------------- -----------
    DV_CONFIGURE_STATUS  TRUE
    DV_ENABLE_STATUS     TRUE
    ```

  - If you want to find the Database Vault status of all PDBs in a multitenant environment, as a common user with administrative privileges, then query CDB_DV_STATUS, which provides the addition of a container ID (CON_ID) field.
- For Oracle Label Security, query the following data dictionary views, which are similar to their Database Vault equivalent views:
  - DBA_OLS_STATUS
  - CDB_OLS_STATUS

Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control

Oracle Enterprise Manager Cloud Control (Cloud Control) provides pages for managing Oracle Database Vault.

The Oracle Database Vault pages can be used to administer and monitor Database Vault-protected databases from a centralized console. This console enables you to automate alerts, view Database Vault reports, and propagate Database Vault policies to other Database Vault-protected databases.

Before you try to log in, ensure that you have configured the Cloud Control target databases that you plan to use with Database Vault by following the Oracle Enterprise Manager online help. Oracle Database Vault must also be registered with the Oracle database.

1. Start Cloud Control.

For example:

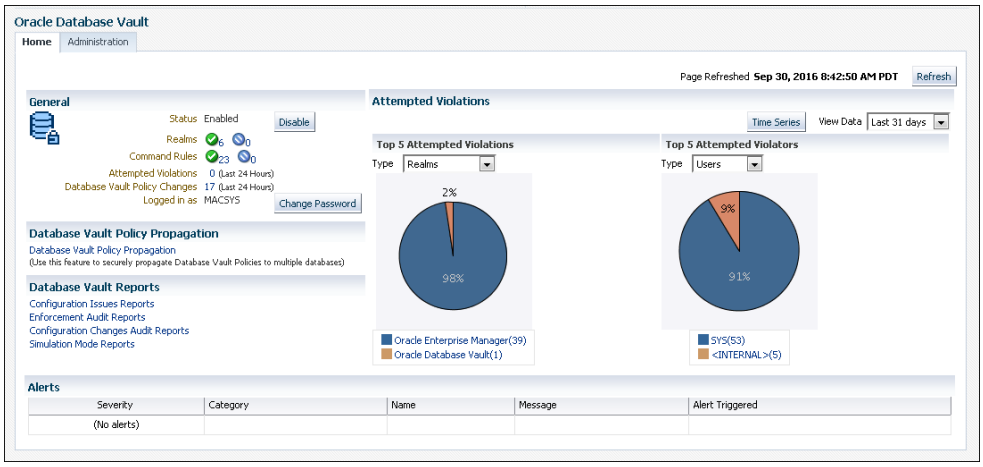
```
https://myserver.example.com:7799/em
```

2. Log in to Cloud Control as a security administrator.
3. In the Cloud Control home page, from the **Targets** menu, select **Databases**.
4. In the Databases page, select the link for the Oracle Database Vault-protected database to which you want to connect.

   The Database home page appears.
5. From the **Security** menu, select **Database Vault**.

   The Database Login page appears.
6. Enter the following information:
   - **Username:** Enter the name of a user who has been granted the appropriate Oracle Database Vault role:
     - Creating and propagating Database Vault policies: `DV_OWNER` or `DV_ADMIN` role, `SELECT ANY DICTIONARY` privilege
     - Viewing Database Vault alerts and reports: `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role, `SELECT ANY DICTIONARY` privilege
   - **Password:** Enter your password.
   - **Role:** Select **NORMAL** from the list.
   - **Save as:** Select this check box if you want these credentials to be automatically filled in for you the next time that this page appears. The credentials are stored in Enterprise Manager in a secured manner. Access to these credentials depends on the user who is currently logged in.

   The Database Vault home page appears.



Related Topics

- About Oracle Database Vault Roles
  Oracle Database Vault provides a set of roles that are required for managing Oracle Database Vault.

- Using Oracle Database Vault with Oracle Enterprise Manager
  Oracle Database Vault administrators can perform tasks in Oracle Enterprise Manager Cloud Control such as propagating polices to other databases.

Quick Start Tutorial: Securing a Schema from DBA Access

This tutorial shows how to create a realm around the HR schema.

- About This Tutorial
  In this tutorial, you create a realm around for the HR sample database schema by using the Oracle Database Vault PL/SQL packages.
- Step 1: Log On as SYSTEM to Access the HR Schema
  You must enable the HR schema for this tutorial.
- Step 2: Create a Realm
  Realms can protect one or more schemas, individual schema objects, and database roles.
- Step 3: Create the SEBASTIAN User Account
  At this stage, there are no database accounts or roles authorized to access or otherwise manipulate the database objects the realm will protect.
- Step 4: Have User SEBASTIAN Test the Realm
  At this stage, have user SEBASTIAN test the realm, even though he has the READ ANY TABLE system privilege.
- Step 5: Create an Authorization for the Realm
  Next, user SEBASTIAN must be granted authorization to the HR Apps realm, so that he can access the HR.EMPLOYEES table.
- Step 6: Test the Realm
  To test the realm, you must try to access the EMPLOYEES table as a user other than HR.
- Step 7: If Unified Auditing Is Not Enabled, Then Run a Report
  Because you enabled auditing on failure for the HR Apps realm, you can generate a report to find any security violations.
- Step 8: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

About This Tutorial

In this tutorial, you create a realm around for the HR sample database schema by using the Oracle Database Vault PL/SQL packages.

In the HR schema, the EMPLOYEES table has information such as salaries that should be hidden from most employees in the company, including those with administrative access. To accomplish this, you add the HR schema to the secured objects of the protection zone, which in Oracle Database Vault is called a *realm*, inside the database. Then you grant limited authorizations to this realm. Afterward, you test the realm to make sure it has been properly secured. And finally, to see how Oracle Database Vault provides an audit trail on suspicious activities like the one you will try when you test the realm, you will run a report.

Step 1: Log On as SYSTEM to Access the HR Schema

You must enable the HR schema for this tutorial.

Before you begin this tutorial, ensure that the HR sample schema is installed. *Oracle Database Sample Schemas* describes how to install the sample schemas.

1. Log into the database instance as a user who has been granted the DBA role, and then access the HR schema.

   For example:

   ```
   sqlplus system
   Enter password: password
   ```

2. In a multitenant environment, connect to the appropriate PDB.

   For example:

   ```
   CONNECT SYSTEM@my_pdb
   Enter password: password
   ```

   To find the available PDBs, run the `show pdbs` command. To check the current PDB, run the `show con_name` command.

3. Query the HR.EMPLOYEES table as follows.

   ```
   SELECT FIRST_NAME, LAST_NAME, SALARY FROM HR.EMPLOYEES WHERE ROWNUM < 10;
   ```

   Output similar to the following appears:

   ```
   FIRST_NAME           LAST_NAME                 SALARY
   -------------------- ------------------------- ----------
   Steven               King                       24000
   Neena                Kochhar                    17000
   Lex                  De Haan                    17000
   Alexander            Hunold                      9000
   Bruce                Ernst                       6000
   David                Austin                      4800
   Valli                Pataballa                   4800
   Diana                Lorentz                     4200
   Nancy                Greenberg                  12008

   9 rows selected.
   ```

4. If the HR schema is locked and expired, log into the database instance as the DV_ACCTMGR user and unlock and unexpire the account. For example:

   ```
   sqlplus bea_dvacctmgr -- For a multitenant environment, sqlplus
   bea_dvacctmgr@hrpdb
   Enter password: password

   ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

   As you can see, SYSTEM has access to the salary information in the EMPLOYEES table of the HR schema. This is because SYSTEM is automatically granted the DBA role, which includes the `SELECT ANY TABLE` system privilege.

5. Do not exit SQL*Plus.

Step 2: Create a Realm

Realms can protect one or more schemas, individual schema objects, and database roles.

Once you create a realm, you can create security restrictions that apply to the schemas and their schema objects within the realm. You will need to create a realm for the HR schema.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the DV_OWNER or DV_ADMIN role and the SELECT ANY DICTIONARY privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Realms**.
3. In the Realms page of Oracle Database Vault Administrator, click **Create**.
4. In the Create Realm page, under General, enter HR Apps after **Name**.
5. In the **Description** field, enter Realm to protect the HR schema.
6. Leave the **Mandatory Realm** check box unchecked.
7. After Status, ensure that **Enabled** is selected so that the realm can be used.
8. Under Audit Options, ensure that **Audit On Failure** is selected so that you can create an audit trial later on.
9. Click **Next** to display the Realm secured objects page.
10. Click the **Add** button and in the Add Secured Object dialog box, enter the following information:
    - **Owner:** Enter HR to select the HR schema.
    - **Object Type:** Enter TABLE.
    - **Object Name:** Enter EMPLOYEES.
11. Click **OK**.

    The HR.EMPLOYEES table is added to the Create Realm : Realm Secured Objects page.
12. Click **Done**, and then click **Finish**.

At this stage, you have created the realm but you have not assigned any authorizations to it. You will take care of that later on in this tutorial.

Step 3: Create the SEBASTIAN User Account

At this stage, there are no database accounts or roles authorized to access or otherwise manipulate the database objects the realm will protect.

So, the next step is to authorize database accounts or database roles so that they can have access to the schemas within the realm. You will create the SEBASTIAN user account.

1. In SQL*Plus, connect as the Database Vault Account Manager, who has the DV_ACCTMGR role, and create the local user SEBASTIAN.

   For example:

```
CONNECT bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
Enter password: password

GRANT CREATE SESSION TO SEBASTIAN IDENTIFIED BY password;
```

Replace *password* with a password that is secure. See *Oracle Database Security Guide* for the minimum requirements for creating passwords.

2. Connect as SYS with the SYSDBA privilege, and then grant SEBASTIAN the following additional privilege.

```
CONNECT SYS AS SYSDBA -- Or, CONNECT SYS@hrpdb AS SYSDBA
Enter password: password

GRANT READ ANY TABLE TO SEBASTIAN;
```

Do not exit SQL*Plus; you will need it for Step 6: Test the Realm, when you test the realm.

Step 4: Have User SEBASTIAN Test the Realm

At this stage, have user SEBASTIAN test the realm, even though he has the READ ANY TABLE system privilege.

1. Connect as user SEBASTIAN.

```
CONNECT sebastian
Enter password: password
```

2. Query the HR.EMPLOYEES table.

```
SELECT COUNT(*) FROM HR.EMPLOYEES;
```

The following output should appear:

```
ERROR at line 1:
ORA-01031: insufficient privileges
```

Even though user SEBASTIAN has the READ ANY TABLE system privilege, he cannot query the HR.EMPLOYEES table, because the HR Apps realm takes precedence over the READ ANY TABLE system privilege.

Step 5: Create an Authorization for the Realm

Next, user SEBASTIAN must be granted authorization to the HR Apps realm, so that he can access the HR.EMPLOYEES table.

1. In the Realms page of Database Vault Administrator, select the **HR Apps** in the list of realms, and then click **Edit**.
2. Click the **Next** button until you reach the Realm authorizations page.
3. Click **Add** and then enter the following information in the Add Authorizations dialog box:
   - **Realm Authorization Grantee:** Enter SEBASTIAN.
   - **Realm Authorization Type:** Select **Participant** from the list.
   - **Realm Authorization Ruleset:** Leave this field blank.
4. Click **OK**.

The Participant authorization allows the user SEBASTIAN in the HR Apps realm to manage access, manipulate, and create objects protected by the HR Apps realm. In this case, the HR user and SEBASTIAN are the only users allowed to view the EMPLOYEES table.

5. Click **Done**, and then **Finish**.

Step 6: Test the Realm

To test the realm, you must try to access the EMPLOYEES table as a user other than HR.

The SYSTEM account normally has access to all objects in the HR schema, but now that you have safeguarded the EMPLOYEES table with Oracle Database Vault, this is no longer the case.

1. In SQL*Plus, connect as SYSTEM.

```
CONNECT SYSTEM -- Or, CONNECT SYSTEM@hrpdb
Enter password: password
```

2. Try accessing the salary information in the EMPLOYEES table again:

```
SELECT FIRST_NAME, LAST_NAME, SALARY FROM HR.EMPLOYEES WHERE ROWNUM <10;
```

The following output should appear:

```
Error at line 1:
ORA-01031: insufficient privileges
```

SYSTEM no longer has access to the salary information in the EMPLOYEES table. (In fact, even user SYS does not have access to this table.) However, user SEBASTIAN does have access to this information.

3. Connect as user SEBASTIAN.

```
CONNECT sebastian -- Or, CONNECT sebastian@hrpdb
Enter password: password
```

4. Perform the following query:

```
SELECT FIRST_NAME, LAST_NAME, SALARY FROM HR.EMPLOYEES WHERE ROWNUM <10;
```

Output similar to the following appears:

```
FIRST_NAME           LAST_NAME                     SALARY
-------------------- ------------------------- ----------
Steven               King                           24000
Neena                Kochhar                        17000
Lex                  De Haan                        17000
Alexander            Hunold                          9000
Bruce                Ernst                           6000
David                Austin                          4800
Valli                Pataballa                       4800
Diana                Lorentz                         4200
Nancy                Greenberg                      12008

9 rows selected.
```

Step 7: If Unified Auditing Is Not Enabled, Then Run a Report

Because you enabled auditing on failure for the HR Apps realm, you can generate a report to find any security violations.

For example, you could generate a report for the violation that you attempted in Step 6: Test the Realm.

1. In SQL*Plus, connect as user SYSTEM and ensure that unified auditing is not enabled.

   ```
   CONNECT SYSTEM -- Or, CONNECT SYSTEM@hrpdb
   Enter password: password

   SQL> SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Unified Auditing';
   ```

   If VALUE returns TRUE, then you cannot complete this section. Go to Step 8: Remove the Components for This Tutorial.

   If unified auditing is enabled, then you must create a unified audit policy to capture events. See *Oracle Database Security Guide* for information about how to create unified audit policies for Oracle Database Vault.

2. In the Database Vault Administrator page, click **Home** to display the home page.
3. In the Database Vault Home page, under **Reports**, select **Database Vault Reports**.
4. In the Database Vault Reports page, select **Database Vault Enforcement Audit Report**.
5. From the **Database Vault Audit Report** list, select **Realm Audit Report**.
6. In the Search area, from the **Command** menu, select **Equals** and in the text field, enter SELECT. Then click **Search**.

   The report appears in the table that follows the Search region.
7. Click **OK** to exit the report.

Oracle Database Vault generates a report listing the type of violation (in this case, the SELECT statement entered in the previous section), when and where it occurred, the login account who tried the violation, and what the violation was.

Step 8: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. Drop user SEBASTIAN.

   In SQL*Plus, log on as the Oracle Database Vault account manager (for example, bea_dvacctmgr) and then drop SEBASTIAN as follows:

   ```
   sqlplus bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
   Enter password: password

   DROP USER SEBASTIAN;
   ```

2. Delete the HR Apps realm.

a. In Cloud Control, ensure that you are logged in as a user who has the `DV_OWNER` role.
b. In the Database Vault Home page, click **Administration**.
c. In the Realms page, select `HR Apps` from the list of realms.
d. Click **Delete**, and in the Confirmation window, click **Yes**.

3. If necessary, in SQL*Plus, lock and expire the `HR` account.

```
ALTER USER HR ACCOUNT LOCK PASSWORD EXPIRE;
```

4

Performing Privilege Analysis to Find Privilege Use

Privilege analysis dynamically analyzes the privileges and roles that users use and do not use.

- What Is Privilege Analysis?
  Oracle Database Vault with Oracle Database Release 12c includes a feature called privilege analysis to help you increase the security of your applications and database operations.
- Creating and Managing Privilege Analysis Policies
  You can create and manage privilege analysis policies in either SQL*Plus or in Enterprise Manager Cloud Control.
- Creating Roles and Managing Privileges Using Cloud Control
  You can create new roles using privileges found in a privilege analysis report and then grant this role to users.
- Tutorial: Using Capture Runs to Analyze ANY Privilege Use
  This tutorial demonstrates how to create capture runs to analyze the use of the `READ ANY TABLE` system privilege.
- Tutorial: Analyzing Privilege Use by a User Who Has the DBA Role
  This tutorial demonstrates how to analyze the privilege use of a user who has the `DBA` role and performs database tuning operations.
- Privilege Analysis Policy and Report Data Dictionary Views
  Oracle Database provides a set of data dictionary views that provide information about analyzed privileges.

What Is Privilege Analysis?

Oracle Database Vault with Oracle Database Release 12c includes a feature called privilege analysis to help you increase the security of your applications and database operations.

- About Privilege Analysis
  Because it is a dynamic analysis, it captures real privileges and roles that were actually used.
- How Privilege Analysis Works with Pre-Compiled Database Objects
  Privilege analysis can be used to capture the privileges that have been exercised on pre-compiled database objects.
- Who Can Perform Privilege Analysis?
  To use privilege analysis, you must be granted the `CAPTURE_ADMIN` role.
- Types of Privilege Analysis
  You can create different types of privilege analysis policies to achieve specific goals.

- Benefits and Use Cases of Privilege Analysis
  Analyzing privilege use is beneficial in finding unnecessarily granted privileges.
- How Does a Multitenant Environment Affect Privilege Analysis?
  You can create and use privilege analysis policies in a multitenant environment.

About Privilege Analysis

Because it is a dynamic analysis, it captures real privileges and roles that were actually used.

Privilege analysis captures privileges used by database users and applications at runtime. If your applications include definer's rights and invoker's rights procedures, then privilege analysis captures the privileges that are required to compile a procedure and execute it, even if the procedure was compiled before the privilege capture was created and enabled.

Running inside the Oracle Database kernel, privilege analysis helps reduce the attack surface of applications and increase operational security by identifying used and unused privileges. Privilege analysis can be used after you install Oracle Database Release 12*c* without any additional configuration steps.

> **Note:**
>
> If you want to configure privilege analysis by using Oracle Enterprise Manager Cloud Control, then ensure that you have the latest plug-in. For information about how to deploy a plug-in, see *Enterprise Manager Cloud Control Administrator's Guide*.

How Privilege Analysis Works with Pre-Compiled Database Objects

Privilege analysis can be used to capture the privileges that have been exercised on pre-compiled database objects.

Examples of these objects are PL/SQL packages, procedures, functions, views, triggers, and Java classes and data.

Because these privileges may not be exercised during run time when a stored procedure is called, these privileges are collected when you generate the results for any database-wide capture, along with run-time captured privileges. A privilege is treated as an unused privilege when it is not used in either pre-compiled database objects or run-time capture, and it is saved under the run-time capture name. If a privilege is used for pre-compiled database objects, then it is saved under the capture name `ORA$DEPENDENCY`. If a privilege is captured during run time, then it is saved under the run-time capture name. If you want to know what the used privileges are for both pre-compiled database objects and run-time usage, then you must query both the `ORA$DEPENDENCY` and run-time captures. For unused privileges, you only need to query with the run-time capture name.

To find a full list of the pre-compiled objects on which privilege analysis can be used, query the `TYPE` column of the `ALL_DEPENDENCIES` data dictionary view.

Who Can Perform Privilege Analysis?

To use privilege analysis, you must be granted the `CAPTURE_ADMIN` role.

You use the `DBMS_PRIVILEGE_CAPTURE` PL/SQL package to manage privilege capture. You use the data dictionary views provided by privilege analysis to analyze your privilege use.

Types of Privilege Analysis

You can create different types of privilege analysis policies to achieve specific goals.

- **Role-based privilege use capture.** You must provide a list of roles. If the roles in the list are enabled in the database session, then the used privileges for the session will be captured. You can capture privilege use for the following types of roles: Oracle default roles, user-created roles, Code Based Access Control (CBAC) roles, and secure application roles.
- **Context-based privilege use capture.** You must specify a Boolean expression only with the `SYS_CONTEXT` function. The used privileges will be captured if the condition evaluates to `TRUE`.
- **Role- and context-based privilege use capture.** You must provide both a list of roles that are enabled and a `SYS_CONTEXT` Boolean expression for the condition. When any of these roles is enabled in a session and the given context condition is satisfied, then privilege analysis starts capturing the privilege use.
- **Database-wide privilege capture.** If you do not specify any type in your privilege analysis policy, then the used privileges in the database will be captured, except those for the user `SYS`. (This is also referred to as unconditional analysis, because it is turned on without any conditions.)

Note the following restrictions:

- You can enable only one privilege analysis policy at a time. The only exception is that you can enable a database-wide privilege analysis policy at the same time as a non-database-wide privilege analysis policy, such as a role or context attribute-driven analysis policy.
- You cannot analyze the privileges of the `SYS` user.
- Privilege analysis shows the grant paths to the privilege but it does not suggest which grant path to keep.
- If the role, user, or object has been dropped, then the values that reflect the privilege captures for these in the privilege analysis data dictionary views are dropped as well.

Benefits and Use Cases of Privilege Analysis

Analyzing privilege use is beneficial in finding unnecessarily granted privileges.

- Unnecessarily Granted Privileges of Applications
  The privileges of the account that accesses a database should only be limited to the privileges that are strictly required by the application.

- Development of Secure Applications
  During the application development phase, some administrators may grant many powerful system privileges and roles to application developers.

Unnecessarily Granted Privileges of Applications

The privileges of the account that accesses a database should only be limited to the privileges that are strictly required by the application.

But when an application is developed, especially by a third party, more privileges than necessary may be granted to the application connection pool accounts for convenience. In addition, some developers grant system and application object privileges to the `PUBLIC` role.

For example, to select from application data and run application procedures, the system privileges `SELECT ANY TABLE` and `EXECUTE ANY PROCEDURE` are granted to an application account `appsys`. The account `appsys` now can access non-application data even if he or she does not intend to. In this situation, you can analyze the privilege usage by user `appsys`, and then based on the results, revoke and grant privileges as necessary.

Development of Secure Applications

During the application development phase, some administrators may grant many powerful system privileges and roles to application developers.

The administrators may do this because at that stage they may not know what privileges the application developer needs.

Once the application is developed and working, the privileges that the application developer needs — and does not need — become more apparent. At that time, the security administrator can begin to revoke unnecessary privileges. However, the application developer may resist this idea on the basis that the application is currently working without problems. The administrator can use privilege analysis to examine each privilege that the application uses, to ensure that when he or she does revoke any privileges, the application will continue to work.

For example, `app_owner` is an application database user through whom the application connects to a database. User `app_owner` must query tables in the `OE`, `SH`, and `PM` schemas. Instead of granting the `SELECT` object privilege on each of the tables in these schemas, a security administrator grants the `SELECT ANY TABLE` privilege to `app_owner`. After a while, a new schema, `HR`, is created and sensitive data are inserted into `HR.EMPLOYEES` table. Because user `app_owner` has the `SELECT ANY TABLE` privilege, he can query this table to access its sensitive data, which is a security issue. Instead of granting system privileges (particularly the `ANY` privileges), it is far better to grant object privileges for specific tables.

How Does a Multitenant Environment Affect Privilege Analysis?

You can create and use privilege analysis policies in a multitenant environment.

If you are using a multitenant environment, then you can create privilege analysis policies in either the CDB root or in individual PDBs. The privilege analysis policy applies only to the container in which it is created, either to the privileges used within the CDB root or the application root, or to the privileges used within a PDB. It cannot be applied globally throughout the multitenant environment. You can grant the

CAPTURE_ADMIN role locally to a local user or a common user. You can grant the CAPTURE_ADMIN role commonly to common users.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for more information about multitenant container databases (CDBs)

Creating and Managing Privilege Analysis Policies

You can create and manage privilege analysis policies in either SQL*Plus or in Enterprise Manager Cloud Control.

- About Creating and Managing Privilege Analysis Policies
  You can use Oracle Enterprise Manager Cloud Control or the DBMS_PRIVILEGE_CAPTURE PL/SQL package to analyze privileges.
- General Steps for Managing Privilege Analysis
  You must follow a general set of steps to analyze privileges.
- Creating a Privilege Analysis Policy
  You can create a privilege analysis policy in either Enterprise Manager Cloud Control or from SQL*Plus, using the DBMS_PRIVILEGE_CAPTURE PL/SQL package.
- Examples of Privilege Analysis Policies
  You can create a variety of privilege analysis policies.
- Enabling a Privilege Analysis Policy
  You can enable a privilege analysis policy using either Enterprise Manager Cloud Control or from SQL*Plus, using the DBMS_PRIVILEGE_CAPTURE PL/SQL package.
- Disabling a Privilege Analysis Policy
  You can disable a privilege analysis policy using either Enterprise Manager Cloud Control or from SQL*Plus, using the DBMS_PRIVILEGE_CAPTURE PL/SQL package.
- Generating a Privilege Analysis Report
  You can generate a privilege analysis policy report using either Enterprise Manager Cloud Control or from SQL*Plus, using the DBMS_PRIVILEGE_CAPTURE PL/SQL package.
- Dropping a Privilege Analysis Policy
  You can drop a privilege analysis policy report using either Enterprise Manager Cloud Control or from SQL*Plus, using the DBMS_PRIVILEGE_CAPTURE PL/SQL package.

About Creating and Managing Privilege Analysis Policies

You can use Oracle Enterprise Manager Cloud Control or the DBMS_PRIVILEGE_CAPTURE PL/SQL package to analyze privileges.

Before you can do so, you must be granted the CAPTURE_ADMIN role. The DBMS_PRIVILEGE_CAPTURE package enables you to create, enable, disable, and drop privilege analysis policies. It also generates reports that show the privilege usage, which you can view in DBA_* views.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for detailed information about the `DBMS_PRIVILEGE_CAPTURE` PL/SQL package

General Steps for Managing Privilege Analysis

You must follow a general set of steps to analyze privileges.

1. Define the privilege analysis policy.
2. Enable the privilege analysis policy.

   This step begins recording the privilege use that the policy defined. Optionally, specify a name for this capture run. Each time you enable a privilege analysis policy, you can create a different capture run for it. In this way, you can create multiple named capture runs for comparison analysis later on.
3. Optionally, enable the policy to capture dependency privileges if you want to capture the privileges that are used by definer's rights and invoker's rights program units.
4. After a sufficient period of time to gather data, disable the privilege analysis policy's recording of privilege use.

   This step stops capturing the privilege use for the policy.
5. Generate privilege analysis results.

   This step writes the results to the data dictionary views described in Privilege Analysis Policy and Report Data Dictionary Views.
6. Optionally, disable and then drop the privilege analysis policy and capture run.

   Dropping a privilege analysis policy deletes the data captured by the policy.

Creating a Privilege Analysis Policy

You can create a privilege analysis policy in either Enterprise Manager Cloud Control or from SQL*Plus, using the `DBMS_PRIVILEGE_CAPTURE` PL/SQL package.

- About Creating a Privilege Analysis Policy
  When a policy is created, it resides in the Oracle data dictionary and the `SYS` schema.
- Creating a Privilege Analysis Policy in Enterprise Manager Cloud Control
  You can create a privilege analysis policy in Cloud Control.
- Creating a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE
  The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` procedure creates a privilege analysis policy.

About Creating a Privilege Analysis Policy

When a policy is created, it resides in the Oracle data dictionary and the `SYS` schema.

However, both `SYS` and the user who created the policy can drop it. After you create the policy, you must manually enable it so that it can begin to analyze privilege use. If

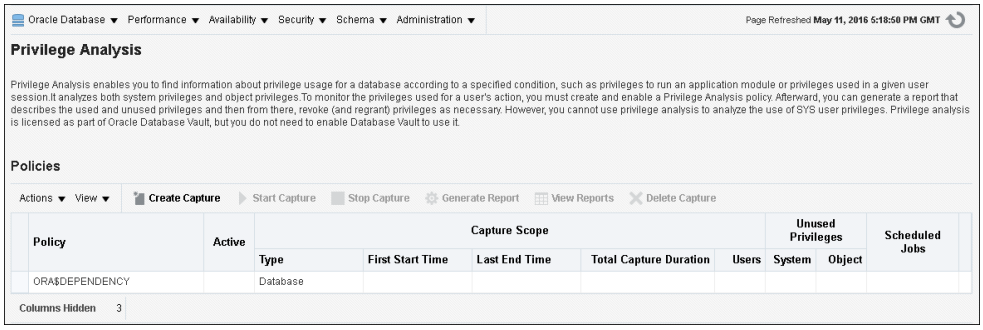
you want to configure privilege analysis by using Oracle Enterprise Manager Cloud Control, then ensure that you have the latest plug-in. For information about how to deploy a plug-in, see *Enterprise Manager Cloud Control Administrator's Guide*.

Creating a Privilege Analysis Policy in Enterprise Manager Cloud Control

You can create a privilege analysis policy in Cloud Control.

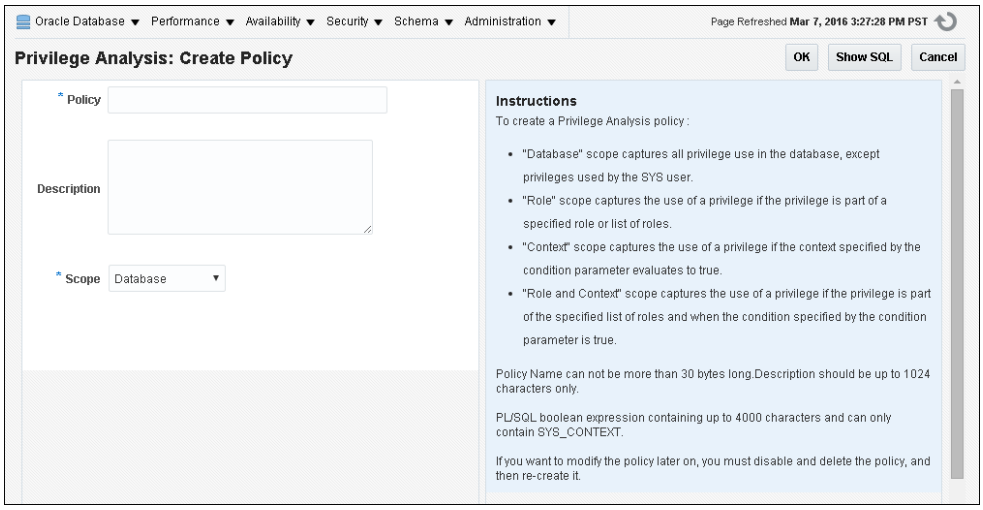
1. Log in to Cloud Control as a user who has been granted the `CAPTURE_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. *Oracle Database 2 Day DBA* explains how to log in.
2. From the **Security** menu, select **Privilege Analysis**.

   The Privilege Analysis home page appears.



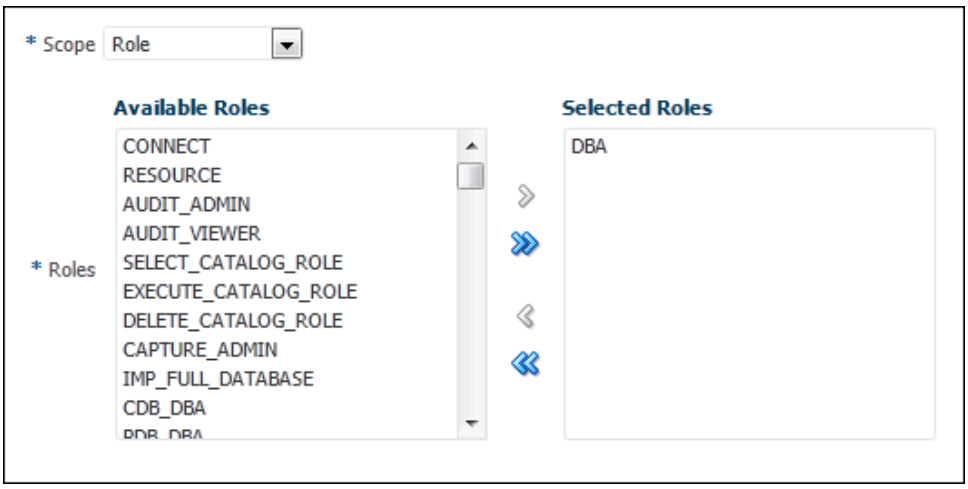
3. In the Privilege Analysis page, under Policies, select **Create Capture**.

   The Privilege Analysis: Create Policy page appears.

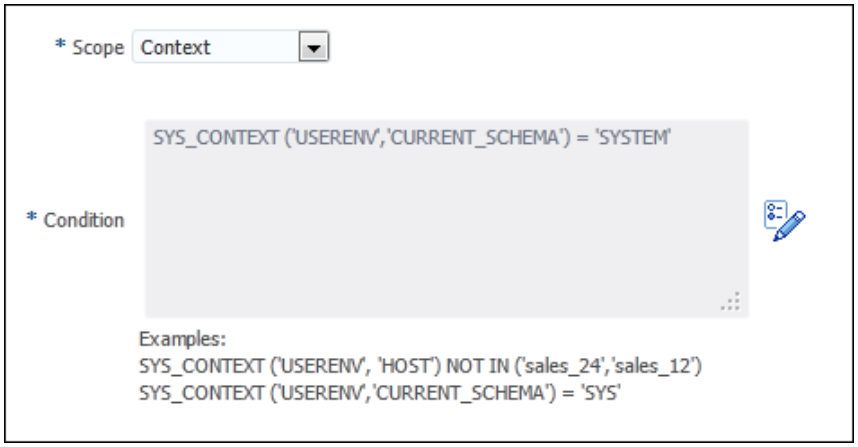


4. Enter the following information:
   - **Policy**: Enter a unique name for the privilege analysis policy. You can find the names of existing policies by querying the `NAME` column of the `DBA_PRIV_CAPTURES` view. You can include spaces in the name and have a maximum of 128 characters in this name.
   - **Description**: Optionally, enter a description for the policy, in up to 1024 characters.

- **Scope**: Select from the following types:
  - **Database** captures all privileges that were used in the entire database, except privileges from user SYS.
  - **Role** captures privileges from one or more roles that you specify. If the roles in the list are enabled in the database session, then the used privileges for the session will be captured. If you select this option, then the Create Policy page displays the **Available Roles** list.



  - **Context** captures privileges when the condition that you specify evaluates to TRUE. If you select this option, then the Capture Policy page displays a **Condition** field. To build the condition, select the edit icon on the right of this field to display the Policy Expression Builder dialog box.



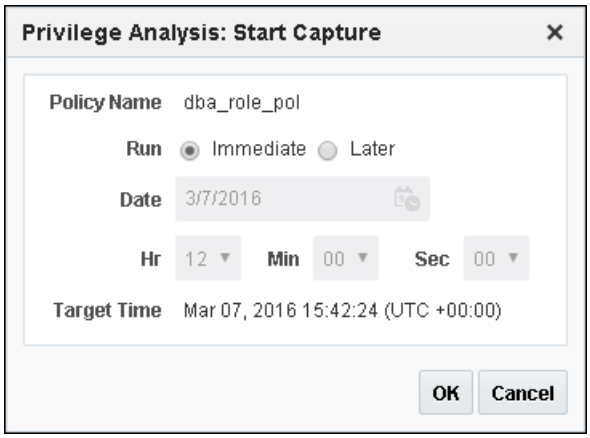
  - **Role and Context** captures privileges from one of the specified roles when the context condition evaluates to TRUE. If you select this option, then both the list of available roles and **Condition** field appear.

5. Click **OK**.

   The new policy appears in the Policies area of the Privilege Analysis page.

6. To enable the policy so that it can begin to capture privilege use, return to the main Privilege Analysis policy page, select the policy under **Policies**, and then click **Start Capture**.

The Privilege Analysis: Start Capture dialog box appears.



7. Enter the following information to set the time in which the capture will begin:
   - To start the privilege capture process immediately, click **Immediate** and then click the **OK** button.
   - To start the privilege capture at a later date, click **Later**, specify the date and time that you want to capture process to begin, and then click **OK**.

   The Privilege Analysis page appears. The time that you set for the policy to begin is listed under **First Start Time** for the policy. If you want to modify the start time, select the policy and click **Start Capture**.

Creating a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE

The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` procedure creates a privilege analysis policy.

After you create the privilege analysis policy, you can find it listed in the `DBA_PRIV_CAPTURES` data dictionary view.

- Use the following syntax for the `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` procedure:

```
DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
   name              VARCHAR2,
   description       VARCHAR2 DEFAULT NULL,
   type              NUMBER DEFAULT DBMS_PRIVILEGE_CAPTURE.G_DATABASE,
   roles             ROLE_NAME_LIST DEFAULT ROLE_NAME_LIST(),
   condition         VARCHAR2 DEFAULT NULL);
```

In this specification:

- `name`: Specifies the name of the privilege analysis policy to be created. Ensure that this name is unique and no more than 128 characters. You can include spaces in the name, but you must enclose the name in single quotation marks whenever you refer to it. To find the names of existing policies, query the `NAME` column of the `DBA_PRIV_CAPTURES` view.
- `description`: Describes the purpose of the privilege analysis policy, up to 1024 characters in mixed-case letters. Optional.
- `type`: Specifies the type of capture condition. If you omit the `type` parameter, then the default is `DBMS_PRIVILEGE_CAPTURE.G_DATABASE`. Optional.

Enter one of the following types:

- `DBMS_PRIVILEGE_CAPTURE.G_DATABASE`: Captures all privileges used in the entire database, except privileges from user `SYS`.
- `DBMS_PRIVILEGE_CAPTURE.G_ROLE`: Captures privileges for the sessions that have the roles enabled. If you enter `DBMS_PRIVILEGE_CAPTURE.G_ROLE` for the `type` parameter, then you must also specify the `roles` parameter. For multiple roles, separate each role name with a comma.
- `DBMS_PRIVILEGE_CAPTURE.G_CONTEXT`: Captures privileges for the sessions that have the condition specified by the `condition` parameter evaluating to `TRUE`. If you enter `DBMS_PRIVILEGE_CAPTURE.G_CONTEXT` for the `type` parameter, then you must also specify the `condition` parameter.
- `DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT`: Captures privileges for the sessions that have the role enabled and the context condition evaluating to `TRUE`. If you enter `DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT` for the `type` parameter, then you must also specify both the `roles` and `condition` parameters.

- `roles`: Specifies the roles whose used privileges will be analyzed. That is, if a privilege from one of the given roles is used, then the privilege will be analyzed. You must specify this argument if you specify `DBMS_PRIVILEGE_CAPTURE.G_ROLE` or `DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT` for the `type` argument. Each role you enter must exist in the database. (You can find existing roles by querying the `DBA_ROLES` data dictionary view.) For multiple roles, use varray type `role_name_list` to enter the role names. You can specify up to 10 roles.

  For example, to specify two roles:

  ```
  roles => role_name_list('role1', 'role2'),
  ```

- `condition`: Specifies a Boolean expression up to 4000 characters. You must specify this argument if you specify `DBMS_PRIVILEGE_CAPTURE.G_CONTEXT` or `DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT` for the `type` argument. Only `SYS_CONTEXT` expressions with relational operators(==, >, >=, <, <=, <>, `BETWEEN`, and `IN`) are permitted in this Boolean expression.

  The `condition` expression syntax is as follows:

  ```
  predicate::= SYS_CONTEXT(namespace, attribute) relop constant_value |
               SYS_CONTEXT(namespace, attribute)
               BETWEEN
               constant_value
               AND constant_value | SYS_CONTEXT(namespace, attribute)
               IN {constant_value (,constant_value)* }

  relop::= = | < | <= | > | >= | <>

  context_expression::= predicate | (context_expression)
               AND (context_expression) | (context_expression)
               OR (context_expression )
  ```

  For example, to use a condition to specify the IP address `192.0.2.1`:

  ```
  condition => 'SYS_CONTEXT(''USERENV'', ''IP_ADDRESS'')=''192.0.2.1''';
  ```

* You can add as many constant values as you need (for example, `IN {constant_value1}`, or `IN {constant_value1, constant_value2, constant_value3}`).

Remember that after you create the privilege analysis policy, you must enable it, as described in Enabling a Privilege Analysis Policy.

Examples of Privilege Analysis Policies

You can create a variety of privilege analysis policies.

- Example: Privilege Analysis of Database-Wide Privileges
  The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` can be used to analyze database-wide privileges.
- Example: Privilege Analysis of Privilege Usage of Two Roles
  The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` procedure can be used to analyze the privilege usage of multiple roles.
- Example: Privilege Analysis of Privileges During SQL*Plus Use
  The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` procedure can be used to capture privileges for analysis.
- Example: Privilege Analysis of PSMITH Privileges During SQL*Plus Access
  The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` can be used to analyze user access when the user is running SQL*Plus.

Example: Privilege Analysis of Database-Wide Privileges

The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` can be used to analyze database-wide privileges.

Example 4-1 shows how to use the `DBMS_PRIVILEGE_CAPTURE` package to create and enable a privilege analysis policy to record all privilege use in the database.

Example 4-1 Privilege Analysis of Database-Wide Privileges

```
BEGIN
 DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  name          => 'db_wide_capture_pol',
  description   => 'Captures database-wide privileges',
  type          => DBMS_PRIVILEGE_CAPTURE.G_DATABASE);
END;
/
EXEC DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ('db_wide_capture_pol');
```

Example: Privilege Analysis of Privilege Usage of Two Roles

The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` procedure can be used to analyze the privilege usage of multiple roles.

Example 4-2 shows how to analyze the privilege usage of two roles.

Example 4-2 Privilege Analysis of Privilege Usage of Two Roles

```
BEGIN
 DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  name          => 'dba_roles_capture_pol',
  description   => 'Captures DBA and LBAC_DBA role use',
  type          => DBMS_PRIVILEGE_CAPTURE.G_ROLE,
  roles         => role_name_list('dba', 'lbac_dba'));
END;
```

```
/
EXEC DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ('dba_roles_capture_pol');
```

Example: Privilege Analysis of Privileges During SQL*Plus Use

The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` procedure can be used to capture privileges for analysis.

Example 4-3 shows how to analyze privileges used to run SQL*Plus.

Example 4-3 Privilege Analysis of Privileges During SQL*Plus Use

```
BEGIN
 DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  name             => 'sqlplus_capture_pol',
  description      => 'Captures privilege use during SQL*Plus use',
  type             => DBMS_PRIVILEGE_CAPTURE.G_CONTEXT,
  condition        => 'SYS_CONTEXT(''USERENV'', ''MODULE'')=''sqlplus''');
END;
/
EXEC DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ('sqlplus_capture_pol');
```

Example: Privilege Analysis of PSMITH Privileges During SQL*Plus Access

The `DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE` can be used to analyze user access when the user is running SQL*Plus.

Example 4-4 shows how to analyze the privileges used by session user `PSMITH` when running SQL*Plus.

Example 4-4 Privilege Analysis of PSMITH Privileges During SQL*Plus Access

```
BEGIN
 DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  name         => 'psmith_sqlplus_analysis_pol',
  description  => 'Analyzes PSMITH role priv use for SQL*Plus module',
  type         => DBMS_PRIVILEGE_CAPTURE.G_CONTEXT,
  condition    => 'SYS_CONTEXT(''USERENV'', ''MODULE'')=''sqlplus''
                  AND SYS_CONTEXT(''USERENV'', ''SESSION_USER'')=''PSMITH''');
END;
/
EXEC DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ('psmith_sqlplus_analysis_pol');
```

Enabling a Privilege Analysis Policy

You can enable a privilege analysis policy using either Enterprise Manager Cloud Control or from SQL*Plus, using the `DBMS_PRIVILEGE_CAPTURE` PL/SQL package.

- About Enabling a Privilege Analysis Policy
  After you create a privilege analysis policy, you must enable it.
- Enabling a Privilege Analysis Policy Using Cloud Control
  You can enable a privilege analysis policy using Cloud Control.
- Enabling a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE
  The `DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE` procedure enables a privilege policy and creates a capture run name for it.

About Enabling a Privilege Analysis Policy

After you create a privilege analysis policy, you must enable it.

When you enable a privilege analysis policy, you can create a named capture run for the policy's findings. The capture run defines a period of time from when the capture is enabled (begun) and when it is disabled (stopped). This way, you can create multiple runs and then compare them when you generate the privilege capture results. Tutorial: Using Capture Runs to Analyze ANY Privilege Use provides an example of how you can create and generate multiple capture runs.

The general process for managing multiple named capture runs is as follows:

1. Create the policy.
2. Enable the policy for the first run.
3. After a period time to collect user behavior data, disable this policy and its run.
4. Generate the results and then query the privilege analysis data dictionary views for information about this capture run.

   If you omit the `run_name` parameter from the `DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT` procedure, then this procedure looks at all records as a whole and then analyzes them.
5. Re-enable the policy for the second run. You cannot create a new capture run if the policy has not been disabled first.
6. After you have collected the user data, disable the policy and the second run.
7. Generate the results.
8. Query the privilege analysis data dictionary views. The results from both capture runs are available in the views. If you only want to show the results of one of the capture runs, then you can regenerate the results and requery the privilege analysis views.

Once enabled, the privilege analysis policy will begin to record the privilege usage when the condition is satisfied. At any given time, only one privilege analysis policy in the database can be enabled. The only exception is that a privilege analysis policy of type `DBMS_PRIVILEGE_CAPTURE.G_DATABASE` can be enabled at the same time with a privilege analysis of a different type.

When you drop a privilege analysis policy, its associated capture runs are dropped as well and are not reflected in the privilege analysis data dictionary views.

Restarting a database does not change the status of a privilege analysis. For example, if a privilege analysis policy is enabled before a database shutdown, then the policy is still enabled after the database shutdown and restart.

Enabling a Privilege Analysis Policy Using Cloud Control

You can enable a privilege analysis policy using Cloud Control.

1. Log in to Cloud Control as a user who has been granted the `CAPTURE_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. *Oracle Database 2 Day DBA* explains how to log in.
2. From the **Security** menu, select **Privilege Analysis**.

3. Under Policies, select the policy that you want to enable.
4. Select the **Start Capture** button.
5. In the Privilege Analysis: Start Capture dialog box, specify a time to begin the privilege analysis policy.

   To run the policy now, select **Immediate**. To run the policy later, select **Later**, and then specify the hour, minute, second, and the time zone for the policy to begin.
6. Click **OK**.

Enabling a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE

The `DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE` procedure enables a privilege policy and creates a capture run name for it.

The run name defines the period of time that the capture took place.

1. Query the `NAME` and `ENABLED` columns of the `DBA_PRIV_CAPTURES` data dictionary view to find the existing privilege analysis policies and whether they are currently enabled.
2. Run the `DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE` procedure to enable the policy and optionally create a name for a capture run.

   For example, to enable the privilege analysis policy `logon_users_analysis`:

```
BEGIN
  DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE (
    name     => 'logon_users_analysis_pol',
    run_name => 'logon_users_04092016');
END;
/
```

Disabling a Privilege Analysis Policy

You can disable a privilege analysis policy using either Enterprise Manager Cloud Control or from SQL*Plus, using the `DBMS_PRIVILEGE_CAPTURE` PL/SQL package.

- About Disabling a Privilege Analysis Policy
  You must disable the privilege analysis policy before you can generate a privilege analysis report.
- Disabling a Privilege Analysis Policy Using Cloud Control
  You can disable a privilege analysis policy using Cloud Control.
- Disabling a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE
  The `DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE` procedure disables a privilege analysis policy.

About Disabling a Privilege Analysis Policy

You must disable the privilege analysis policy before you can generate a privilege analysis report.

After you disable the policy, then the privileges are no longer recorded. Disabling a privilege analysis policy takes effect immediately for user sessions logged on both before and after the privilege analysis policy is disabled.

Disabling a Privilege Analysis Policy Using Cloud Control

You can disable a privilege analysis policy using Cloud Control.

1. Log in to Cloud Control as a user who has been granted the `CAPTURE_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. *Oracle Database 2 Day DBA* explains how to log in.
2. From the **Security** menu, select **Privilege Analysis**.
3. Under Policies, select the policy that you want to disable.
4. Select **Stop Capture**.
5. In the Privilege Analysis: Stop Capture dialog box, do the following: specify a time to stop the privilege analysis policy.
   a. To stop the policy now, select **Immediate**. To stop the policy later, select **Later**, and then specify the hour, minute, second, and the time zone for the policy to stop.
   b. To generate a report, click the **Generate Report** button. You can view the reports from the Privilege Analysis page by selecting the policy and clicking **View Reports**.
6. Click **OK**.

Disabling a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE

The `DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE` procedure disables a privilege analysis policy.

1. Query the `NAME` and `ENABLED` columns of the `DBA_PRIV_CAPTURES` data dictionary view to find the existing privilege analysis policies and whether they are currently disabled.
2. Run the `DBMS_PRIVILEGE_CAPTURE.DISBLE_CAPTURE` procedure to enable the policy.

   For example, to disable the privilege analysis policy `logon_users_analysis`:

   ```
   EXEC DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ('logon_users_analysis_pol');
   ```

Generating a Privilege Analysis Report

You can generate a privilege analysis policy report using either Enterprise Manager Cloud Control or from SQL*Plus, using the `DBMS_PRIVILEGE_CAPTURE` PL/SQL package.

- About Generating a Privilege Analysis Report
  After the privilege analysis policy has been disabled, you can generate a report.
- Generating a Privilege Analysis Report Using Cloud Control
  You can generate a privilege analysis report using Cloud Control.
- Accessing Privilege Analysis Reports Using Cloud Control
  A privilege analysis report provides information about both used and unused privileges.

- Generating a Privilege Analysis Report Using DBMS_PRIVILEGE_CAPTURE
  The `DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT` procedure generates a report showing the results of a privilege capture.

About Generating a Privilege Analysis Report

After the privilege analysis policy has been disabled, you can generate a report.

In Enterprise Manager Cloud Control, you can view the reports from the Privilege Analysis page **Actions** menu, and from there, revoke and regrant roles and privileges as necessary. To view the report results in SQL*Plus, query the data dictionary views in Privilege Analysis Policy and Report Data Dictionary Views. If a privilege is used during the privilege analysis process and then revoked before you generate the report, then the privilege is still reported as a used privilege, but without the privilege grant path.

Generating a Privilege Analysis Report Using Cloud Control

You can generate a privilege analysis report using Cloud Control.

1. Log in to Cloud Control as a user who has been granted the `CAPTURE_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. *Oracle Database 2 Day DBA* explains how to log in.
2. From the **Security** menu, select **Privilege Analysis**.
3. Under Policies, select the policy whose report you want to generate.
4. Select **Generate Report**.
5. In the Privilege Analysis: Generate Report dialog box, specify a time to generate the report.

   To generate the report now, select **Immediate**. To generate the report later, select **Later**, and then specify the hour, minute, second, and the time zone for the report to generate.
6. Click **OK**.

   In the Privilege Analysis page, a Confirmation message notifies you that a report has been submitted. You can refresh the page until the job is complete. To view the report, select the policy name and then click **View Reports**.

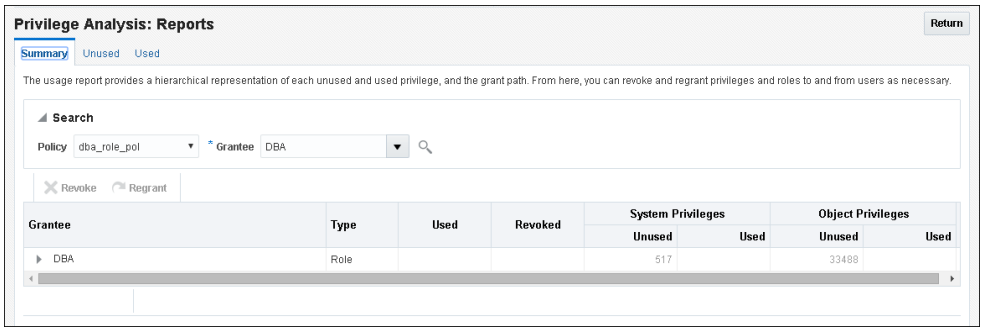
Accessing Privilege Analysis Reports Using Cloud Control

A privilege analysis report provides information about both used and unused privileges.

1. Generate the privilege analysis report.

   See Generating a Privilege Analysis Report Using Cloud Control for more information.
2. In the Privilege Analysis page, select the policy on which you generated a report.
3. Select **View Reports**.

   The Privilege Analysis Reports page appears.



4. To view the report, do the following:

   - By default, the selected report will appear, but to search for a report for another policy, use the Search region to find a different report, or to select a different grantee for the currently selected policy.
   - To view unused privileges, select the **Unused** tab; to view the used privileges, select **Used**. To view a summary of both, select **Summary**.

   From here, you can select roles to revoke or regrant to users as necessary. To do so, under **Grantee**, select the role and then click **Revoke** or **Regrant**.

Generating a Privilege Analysis Report Using DBMS_PRIVILEGE_CAPTURE

The `DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT` procedure generates a report showing the results of a privilege capture.

1. Query the `NAME` and `ENABLED` columns of the `DBA_PRIV_CAPTURES` data dictionary view to find the existing privilege analysis policies and whether they are currently disabled.

   The privilege analysis policy must be disabled before you can generate a privilege analysis report on it.

2. Run the `DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT` procedure using the following syntax:

```
DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT(
  name         VARCHAR2,
  run_name     VARCHAR2 DEFAULT NULL,
  dependency   BOOLEAN DEFAULT NULL);
```

   In this specification:

   - `name`: Specifies the name of the privilege analysis policy. The `DBA_PRIV_CAPTURES` data dictionary view lists the names of existing policies.
   - `run_name`: Specifies the name for the run name for the privilege capture that must be computed. If you omit this setting, then all runs for the given privilege capture are computed.
   - `dependency`: Enter `Y` (yes) or `N` (no) to specify whether the PL/SQL computation privilege usage should be included in the report.

   For example, to generate a report for the privilege analysis policy `logon_users_analysis`:

```
EXEC DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ('logon_users_analysis');
```

3. Query the used privileges from DBA_USED_* data dictionary views with privilege grant paths.

Dropping a Privilege Analysis Policy

You can drop a privilege analysis policy report using either Enterprise Manager Cloud Control or from SQL*Plus, using the DBMS_PRIVILEGE_CAPTURE PL/SQL package.

- About Dropping a Privilege Analysis Policy
  Before you can drop a privilege analysis policy, you must first disable it.
- Dropping a Privilege Analysis Policy Using Cloud Control
  You can drop a privilege analysis policy by using Cloud Control.
- Dropping a Privilege Analysis Policy Using the DBMS_PRIVILEGE_CAPTURE Package
  The DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE procedure drops a privilege analysis policy.

About Dropping a Privilege Analysis Policy

Before you can drop a privilege analysis policy, you must first disable it.

Dropping a privilege analysis policy also drops all the used and unused privilege records associated with this privilege analysis. If you created capture runs for the policy, they are dropped when you drop the policy.

Dropping a Privilege Analysis Policy Using Cloud Control

You can drop a privilege analysis policy by using Cloud Control.

1. Log in to Cloud Control as a user who has been granted the CAPTURE_ADMIN role and the SELECT ANY DICTIONARY privilege. *Oracle Database 2 Day DBA* explains how to log in.
2. From the **Security** menu, select **Privilege Analysis**.
3. Under Policies, select the policy that you want to drop.
4. Select **Delete Capture**.
5. In the Confirmation dialog box, select **Yes**.

Dropping a Privilege Analysis Policy Using the DBMS_PRIVILEGE_CAPTURE Package

The DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE procedure drops a privilege analysis policy.

1. Query the NAME and ENABLE columns of the DBA_PRIV_CAPTURES data dictionary view to find the policy and to check if it is enabled or disabled.
2. If the policy is enabled, then disable it.

   For example:

   ```
   EXEC DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ('logon_users_analysis_pol');
   ```

3. Run the DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE procedure to drop the policy.

For example:

```
EXEC DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE ('logon_users_analysis_pol');
```

If you had enabled the policy with a capture run, then the capture run is dropped as well. To individually drop a capture run, you can run the `DBMS_PRIVILEGE_CAPTURE.DELETE_RUN` procedure, but the policy must exist before you can run this statement.

Creating Roles and Managing Privileges Using Cloud Control

You can create new roles using privileges found in a privilege analysis report and then grant this role to users.

- Creating a Role from a Privilege Analysis Report in Cloud Control
  You can use the report summary to find the least number of privileges an application needs, and encapsulate these privileges into a role.
- Revoking and Regranting Roles and Privileges Using Cloud Control
  You can use Enterprise Manager Cloud Control to revoke and regrant roles and privileges to users.
- Generating a Revoke or Regrant Script Using Cloud Control
  You can generate a script that revokes or regrants privileges from and to users, based on the results of privilege analysis reports.

Creating a Role from a Privilege Analysis Report in Cloud Control

You can use the report summary to find the least number of privileges an application needs, and encapsulate these privileges into a role.

1. Log in to Cloud Control as a user who has been granted the `CAPTURE_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. *Oracle Database 2 Day DBA* explains how to log in.
2. On the Privilege Analysis page, select the policy name, and then from **Actions** menu, click **Create Role**.
3. On the Create Role page, provide the following details, and then click **OK**:
   - Select the policy from which you would like to create a new role.
   - Enter a unique name for the new role that you want to create.
   - Select the **Used** or **Unused** check box, depending on what your role must encapsulate. The role can have used or unused system and object privileges and roles.
   - Select the corresponding radio buttons for **Directly Granted System Privileges**, **Directly Granted Object Privileges**, and **Directly Granted Roles**.

     For example, if you select the **Used** check box, and select:

     – **All** system privileges, then all the used system privileges captured are included in the new role that you are creating.

- **None** for role, then no role that is captured in the policy will be used in the new role.
- **Customize** object privileges, then a list of available used objects privileges captured are displayed, you need to select the privileges from the list to assign to the role.

Revoking and Regranting Roles and Privileges Using Cloud Control

You can use Enterprise Manager Cloud Control to revoke and regrant roles and privileges to users.

1. If Oracle Database Vault is enabled, then ensure that you are authorized as an owner of the Oracle System Privilege and Role Management realm.

   In SQL*Plus, a user who has been granted the `DV_OWNER` role can check the authorization by querying the `DBA_DV_REALM_AUTH` data dictionary view. To grant the user authorization, use the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure.
2. Generate the privilege analysis report.

   See Generating a Privilege Analysis Report Using Cloud Control for more information.
3. In the Privilege Analysis page, select the policy on which you generated a report.
4. Select **View Reports**.
5. In the Privilege Analysis: Reports page, select the **Summary** tab.
6. Under Search, ensure that the **Policy** and **Grantee** menu options are set.
7. Under the Grantee area, expand the grantee options.

   For example, for a role privilege analysis report for a role called `HR_ADMIN` role, you would expand the `HR_ADMIN` role to show the privileges that are associated with it.
8. Select each privilege to revoke and then click **Revoke**, or select **Regrant** to regrant the privilege to the role.

Generating a Revoke or Regrant Script Using Cloud Control

You can generate a script that revokes or regrants privileges from and to users, based on the results of privilege analysis reports.

- About Generating Revoke and Regrant Scripts
  You can perform a bulk revoke of unused system and object privileges and roles by using scripts that you can download after you have generated the privilege analysis.
- Generating a Revoke Script
  You can use Enterprise Manager Cloud Control to generate a script that revokes privileges from users.
- Generating a Regrant Script
  You can use Enterprise Manager Cloud Control to generate a script that regrants privileges that have been revoked from users.

About Generating Revoke and Regrant Scripts

You can perform a bulk revoke of unused system and object privileges and roles by using scripts that you can download after you have generated the privilege analysis.

Later on, if you want to regrant these privileges back to the user, you can generate a regrant script. In order to generate the regrant script, you must have a corresponding revoke script.

Execute the revoke scripts in a development or test environment. Be aware that you cannot revoke privileges and roles from Oracle-supplied accounts and roles.

Generating a Revoke Script

You can use Enterprise Manager Cloud Control to generate a script that revokes privileges from users.

1. If Oracle Database Vault is enabled, then ensure that you are authorized as an owner of the Oracle System Privilege and Role Management realm.

   In SQL*Plus, a user who has been granted the `DV_OWNER` role can check the authorization by querying the `DBA_DV_REALM_AUTH` data dictionary view. To grant the user authorization, use the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure.

2. In Enterprise Manager, access the target Database home page as a user who has been granted the `CAPTURE_ADMIN` role and the `SELECT ANY DICTIONARY` privilege.

   See *Oracle Database 2 Day DBA* for more information.

3. From the **Security** menu, select **Privilege Analysis**.

4. Ensure that the reports you want have been generated.

   See Generating a Privilege Analysis Report Using Cloud Control for more information.

5. In the Privilege Analysis page, from the **Actions** menu, select **Revoke Scripts**.

6. On the Revoke Scripts page, click **Generate**.

   The generate revoke script details wizard is displayed.

7. In the Script Details page, do the following: select a policy name from the **Policy Name** menu against which the revoke script needs to be prepared.

8. In the **Script Name** field, enter a unique name and for **Description**, a description for the script.

   For example, if you want to revoke all the unused privileges, select the **All** option for all the unused privileges and roles, and click **Next**.

   Based on your selection, and the available privileges, all the unused system privileges, object privileges, and roles that are going to be revoked are displayed on the respective pages.

9. For **Grantee (user/role)**, select **All** or **Customize**.

10. Select **All**, **None**, or **Customize** for the **Unused System Privileges**, **Unused Object Privileges**, and **Unused Roles** settings.

11. Click **Next**.

The next pages that appear depend on your selections of **All**, **None**, or **Customize**. If you selected all, the page displays a listing of the privileges. If you selected **None**, the page is bypassed. If you selected **Customize**, then you can individually select the privileges to revoke. The last page that appears is the Review page.

12. Click **Save**.

    The Revoke Scripts page appears.

13. In the Revoke Scripts page, select the newly created SQL script, and then click **Download Revoke Script** to download this script, which contains `REVOKE` SQL statements for each privilege or role.

    To view the script, click the **View Revoke Script** button.

14. To return to the Privilege Analysis page, click **Return**.

Generating a Regrant Script

You can use Enterprise Manager Cloud Control to generate a script that regrants privileges that have been revoked from users.

1. If Oracle Database Vault is enabled, then ensure that you are authorized as an owner of the Oracle System Privilege and Role Management realm.

   In SQL*Plus, a user who has been granted the `DV_OWNER` role can check the authorization by querying the `DBA_DV_REALM_AUTH` data dictionary view. To grant the user authorization, use the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure.

2. In Enterprise Manager, access the target Database home page as a user who has been granted the `CAPTURE_ADMIN` role and the `SELECT ANY DICTIONARY` privilege.

   See *Oracle Database 2 Day DBA* for more information.

3. From the **Security** menu, select **Privilege Analysis**.

4. Ensure that the reports you want have been generated.

   See Generating a Privilege Analysis Report Using Cloud Control for more information.

5. In the Privilege Analysis page, select the policy on which the revoke script was based.

6. From the **Actions** menu, select **Revoke Scripts**.

7. In the Revoke Scripts page, select the policy name that you had created earlier, and then click **Download Regrant Script** to download this script.

   You can view the scripts that are associated with the policy by selecting the **View Revoke Script** and **View Regrant Script** buttons.

Tutorial: Using Capture Runs to Analyze ANY Privilege Use

This tutorial demonstrates how to create capture runs to analyze the use of the `READ ANY TABLE` system privilege.

- Step 1: Create User Accounts
  You must create two users, one user to create the policy and a second user whose privilege use will be analyzed.

- Step 2: Create and Enable a Privilege Analysis Policy
  The user pa_admin must create and enable the privilege analysis policy.
- Step 3: Use the READ ANY TABLE System Privilege
  User app_user uses the READ ANY TABLE system privilege.
- Step 4: Disable the Privilege Analysis Policy
  You must disable the policy before you can generate a report that captures the actions of user app_user.
- Step 5: Generate and View a Privilege Analysis Report
  With the privilege analysis policy disabled, user pa_admin then can generate and view a privilege analysis report.
- Step 6: Create a Second Capture Run
  Next, you are ready to create a second capture run for the ANY_priv_analysis_pol privilege analysis policy.
- Step 7: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

Step 1: Create User Accounts

You must create two users, one user to create the policy and a second user whose privilege use will be analyzed.

1. Log into the database instance as a user who has been granted the DV_ACCTMGR role.

   For example:

   ```
   sqlplus bea_dvacctmgr
   Enter password: password
   ```

   In a multitenant environment, you must connect to the appropriate pluggable database (PDB).

   For example:

   ```
   sqlplus bea_dvacctmgr@hrpdb
   Enter password: password
   ```

   To find the available PDBs, query the DBA_PDBS data dictionary view. To check the current PDB, run the show con_name command.

   If Oracle Database Vault is not enabled, then log into the database instance as a user who has the CREATE USER system privilege.

2. Create the following users:

   ```
   CREATE USER pa_admin IDENTIFIED BY password;
   CREATE USER app_user IDENTIFIED BY password;
   ```

3. Connect as a user who has the privileges to grant roles and system privileges to other users, and who has been granted the owner authorization for the Oracle System Privilege and Role Management realm. (User SYS has these privileges by default.)

   For example:

   ```
   CONNECT dba_psmith -- Or, CONNECT dba_psmith@hrpdb
   Enter password: password
   ```

In SQL*Plus, a user who has been granted the DV_OWNER role can check the authorization by querying the DBA_DV_REALM_AUTH data dictionary view. To grant the user authorization, use the DBMS_MACADM.ADD_AUTH_TO_REALM procedure.

4. Grant the following role and privilege to the users.

```
GRANT CREATE SESSION, CAPTURE_ADMIN TO pa_admin;
GRANT CREATE SESSION, READ ANY TABLE TO app_user;
```

User pa_admin will create the privilege analysis policy that will analyze the READ ANY TABLE query that user app_user will perform.

Step 2: Create and Enable a Privilege Analysis Policy

The user pa_admin must create and enable the privilege analysis policy.

1. Connect as user pa_admin.

```
CONNECT pa_admin -- Or, CONNECT pa_admin@hrpdb
Enter password: password
```

2. Create the following privilege analysis policy:

```
BEGIN
 DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  name           => 'ANY_priv_analysis_pol',
  description    => 'Analyzes system privilege use',
  type           => DBMS_PRIVILEGE_CAPTURE.G_CONTEXT,
  condition      => 'SYS_CONTEXT(''USERENV'', ''SESSION_USER'')=''APP_USER''');
END;
/
```

In this example:

- type specifies the type of capture condition that is defined by the condition parameter, described next. In this policy, the type is a context-based condition.
- condition specifies condition using a Boolean expression that must evaluate to TRUE for the policy to take effect. In this case, the condition checks if the session user is app_user.

3. Enable the policy and create a capture run for it.

```
BEGIN
  DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE (
   name       => 'ANY_priv_analysis_pol',
   run_name   => 'ANY_priv_pol_run_1');
END;
/
```

At this point, the policy is ready to start recording the actions of user app_user.

Step 3: Use the READ ANY TABLE System Privilege

User app_user uses the READ ANY TABLE system privilege.

1. Connect as user app_user.

```
CONNECT app_user -- Or, CONNECT app_user@hrpdb
Enter password: password
```

2. Query the HR.EMPLOYEES table.

```
SELECT FIRST_NAME, LAST_NAME, SALARY FROM HR.EMPLOYEES WHERE SALARY > 12000
ORDER BY SALARY DESC;

FIRST_NAME           LAST_NAME                 SALARY
-------------------- ------------------------- ----------
Steven               King                       24000
Neena                Kochhar                    17000
Lex                  De Haan                    17000
John                 Russell                    14000
Karen                Partners                   13500
Michael              Hartstein                  13000
Shelley              Higgins                    12008
Nancy                Greenberg                  12008
```

Step 4: Disable the Privilege Analysis Policy

You must disable the policy before you can generate a report that captures the actions of user `app_user`.

1. Connect as user `pa_admin`.

   ```
   CONNECT pa_admin -- Or, CONNECT pa_admin@hrpdb
   Enter password: password
   ```

2. Disable the `ANY_priv_analysis_pol` privilege policy.

   ```
   EXEC DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ('ANY_priv_analysis_pol');
   ```

Step 5: Generate and View a Privilege Analysis Report

With the privilege analysis policy disabled, user `pa_admin` then can generate and view a privilege analysis report.

1. As user `pa_admin`, generate the privilege analysis results.

   ```
   BEGIN
     DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT (
       name      => 'ANY_priv_analysis_pol',
       run_name  => 'ANY_priv_pol_run_1');
   END;
   /
   ```

   The generated results are stored in the privilege analysis data dictionary views, which are described in Privilege Analysis Policy and Report Data Dictionary Views.

2. Enter the following commands to format the data dictionary view output:

   ```
   col username format a10
   col sys_priv format a16
   col object_owner format a13
   col object_name format a23
   col run_name format a27
   ```

3. Find the system privileges that `app_user` used and the objects on which he used them during the privilege analysis period.

   ```
   SELECT SYS_PRIV, OBJECT_OWNER, OBJECT_NAME, RUN_NAME FROM DBA_USED_PRIVS WHERE
   USERNAME = 'APP_USER';
   ```

   Output similar to the following appears. The first row shows that `app_user` used the `READ ANY TABLE` privilege on the `HR.EMPLOYEES` table.

```
SYS_PRIV         OBJECT_OWNER  OBJECT_NAME             RUN_NAME
---------------- ------------- ----------------------- ------------------
                 SYSTEM        PRODUCT_PRIVS           ANY_PRIV_POL_RUN_1
                 SYS           DUAL                    ANY_PRIV_POL_RUN_1
                 SYS           DUAL                    ANY_PRIV_POL_RUN_1
CREATE SESSION                                         ANY_PRIV_POL_RUN_1
                 SYS           DBMS_APPLICATION_INFO   ANY_PRIV_POL_RUN_1
READ ANY TABLE   HR            EMPLOYEES               ANY_PRIV_POL_RUN_1
```

At this stage, the privilege analysis results remain available in the privilege analysis data dictionary views, even if you create additional capture runs in the future.

Step 6: Create a Second Capture Run

Next, you are ready to create a second capture run for the `ANY_priv_analysis_pol` privilege analysis policy.

1. As user `pa_admin`, enable the `ANY_priv_analysis_pol` privilege analysis policy to use capture run `ANY_priv_pol_run_1`.

   ```
   BEGIN
     DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE (
      name       => 'ANY_priv_analysis_pol',
      run_name   => 'ANY_priv_pol_run_2');
   END;
   /
   ```

2. Connect as user `app_user`.

   ```
   CONNECT app_user -- Or, CONNECT app_user@hrpdb
   Enter password: password
   ```

3. Query the `HR.JOBS` table.

   ```
   SELECT MAX_SALARY FROM HR.JOBS WHERE MAX_SALARY > 20000;
   ```

4. Connect as user `pa_admin`.

   ```
   CONNECT pa_admin -- Or, CONNECT pa_admin@hrpdb
   Enter password: password
   ```

5. Disable the `ANY_priv_analysis_pol` privilege policy.

   ```
   EXEC DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ('ANY_priv_analysis_pol');
   ```

6. Generate a second privilege analysis report.

   ```
   BEGIN
     DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT (
       name      => 'ANY_priv_analysis_pol',
       run_name  => 'ANY_priv_pol_run_2');
   END;
   /
   ```

7. Find the system privileges that `app_user` used and the objects on which he used them during the privilege analysis period.

   ```
   SELECT SYS_PRIV, OBJECT_OWNER, OBJECT_NAME, RUN_NAME FROM DBA_USED_PRIVS WHERE
   USERNAME = 'APP_USER' ORDER BY RUN_NAME;
   ```

   Output similar to the following appears, which now shows the results of both of the capture runs that user `pa_admin` created.

```
SYS_PRIV         OBJECT_OWNER  OBJECT_NAME             RUN_NAME
---------------- ------------ ---------------------- ----------------------
READ ANY TABLE   HR           EMPLOYEES              ANY_PRIV_POL_RUN_1
                 SYS          DUAL                   ANY_PRIV_POL_RUN_1
CREATE SESSION                                       ANY_PRIV_POL_RUN_1
                 SYS          DUAL                   ANY_PRIV_POL_RUN_1
                 SYSTEM       PRODUCT_PRIVS          ANY_PRIV_POL_RUN_1
                 SYS          DBMS_APPLICATION_INFO  ANY_PRIV_POL_RUN_1
                 SYS          DUAL                   ANY_PRIV_POL_RUN_2
                 SYS          DBMS_APPLICATION_INFO  ANY_PRIV_POL_RUN_2
                 SYSTEM       PRODUCT_PRIVS          ANY_PRIV_POL_RUN_2
                 SYS          DUAL                   ANY_PRIV_POL_RUN_2
READ ANY TABLE   HR           JOBS                   ANY_PRIV_POL_RUN_2
```

Step 7: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. As user `pa_admin`, drop the `ANY_priv_analysis_pol` privilege analysis policy and its associated capture runs.

   ```
   EXEC DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE ('ANY_priv_analysis_pol');
   ```

   Any capture runs that are associated with this policy are dropped automatically when you run the `DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE` procedure.

   Even though in the next steps you will drop the `pa_admin` user, including any objects created in this user's schema, you must manually drop the `ANY_priv_analysis_pol` privilege analysis policy because this object resides in the `SYS` schema.

2. Connect as the user who created the user accounts. If Oracle Database Vault is enabled, then connect as the Oracle Database Vault Account Manager.

   For example:

   ```
   CONNECT bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
   Enter password: password
   ```

3. Drop the users `pa_admin` and `app_user`.

   ```
   DROP USER pa_admin;
   DROP USER app_user;
   ```

Tutorial: Analyzing Privilege Use by a User Who Has the DBA Role

This tutorial demonstrates how to analyze the privilege use of a user who has the `DBA` role and performs database tuning operations.

- Step 1: Create User Accounts
  You must create two users, one to create the privilege analysis policy and a second user whose privilege use will be analyzed.
- Step 2: Create and Enable a Privilege Analysis Policy
  User `pa_admin` must create the and enable the privilege analysis policy.

- Step 3: Perform the Database Tuning Operations
  User `tjones` uses the `DBA` role to perform database tuning operations.
- Step 4: Disable the Privilege Analysis Policy
  You must disable the policy before you can generate a report that captures the actions of user `tjones`.
- Step 5: Generate and View Privilege Analysis Reports
  With the privilege analysis policy disabled, user `pa_admin` can generate and view privilege analysis reports.
- Step 6: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

Step 1: Create User Accounts

You must create two users, one to create the privilege analysis policy and a second user whose privilege use will be analyzed.

1. Log into the database instance as a user who has been granted the `DV_ACCTMGR` role.

   For example:

   ```
   sqlplus bea_dvacctmgr
   Enter password: password
   ```

   In a multitenant environment, you must log into the appropriate pluggable database (PDB).

   For example:

   ```
   sqlplus bea_dvacctmgr@hrpdb
   Enter password: password
   ```

   To find the available PDBs, query the `DBA_PDBS` data dictionary view. To check the current PDB, run the `show con_name` command.

   If Oracle Database Vault is not enabled, then log into the database instance as a user who has the `CREATE USER` system privilege.

2. Create the following users:

   ```
   CREATE USER pa_admin IDENTIFIED BY password;
   CREATE USER tjones IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

3. Connect as a user who has the privileges to grant roles and system privileges to other users, and who has been granted the owner authorization for the Oracle System Privilege and Role Management realm. (User `SYS` has these privileges by default.)

   For example:

   ```
   CONNECT dba_psmith -- Or, CONNECT dba_psmith@hrpdb
   Enter password: password
   ```

In SQL*Plus, a user who has been granted the DV_OWNER role can check the authorization by querying the DBA_DV_REALM_AUTH data dictionary view. To grant the user authorization, use the DBMS_MACADM.ADD_AUTH_TO_REALM procedure.

4. Grant the following roles and privileges to the users.

```
GRANT CREATE SESSION, CAPTURE_ADMIN TO pa_admin;
GRANT CREATE SESSION, DBA TO tjones;
```

User pa_admin will create the privilege analysis policy that will analyze the database tuning operations that user tjones will perform.

Step 2: Create and Enable a Privilege Analysis Policy

User pa_admin must create the and enable the privilege analysis policy.

1. Connect as user pa_admin.

```
CONNECT pa_admin -- Or, CONNECT pa_admin@hrpdb
Enter password: password
```

If Oracle Database Vault is enabled, then log in as the Database Vault Account Manager, who has the DV_ACCTMGR role. Ensure that you are the owner of the Oracle System Privilege and Role Management realm.

2. Create the following privilege analysis policy:

```
BEGIN
 DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  name             => 'dba_tuning_priv_analysis_pol',
  description      => 'Analyzes DBA tuning privilege use',
  type             => DBMS_PRIVILEGE_CAPTURE.G_CONTEXT,
  condition        => 'SYS_CONTEXT(''USERENV'', ''SESSION_USER'')=''TJONES''');
END;
/
```

In this example:

- type specifies the type of capture condition that is defined by the condition parameter, described next. In this policy, the type is a context-based condition.
- condition specifies condition using a Boolean expression that must evaluate to TRUE for the policy to take effect. In this case, the condition checks if the session user is tjones.

3. Enable the policy.

```
EXEC DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE ('dba_tuning_priv_analysis_pol');
```

At this point, the policy is ready to start recording the actions of user tjones.

Step 3: Perform the Database Tuning Operations

User tjones uses the DBA role to perform database tuning operations.

1. Connect as user tjones.

```
CONNECT tjones -- Or, CONNECT tjones@hrpdb
Enter password: password
```

2. Run the following script to create the PLAN_TABLE table.

```
@$ORACLE_HOME/rdbms/admin/utlxplan.sql
```

The location of this script may vary depending on your operating system. This script creates the `PLAN_TABLE` table in the `tjones` schema.

3. Run the following `EXPLAIN PLAN` SQL statement on the `HR.EMPLOYEES` table:

```
EXPLAIN PLAN
 SET STATEMENT_ID = 'Raise in Tokyo'
 INTO PLAN_TABLE
 FOR UPDATE HR.EMPLOYEES
 SET SALARY = SALARY * 1.10
 WHERE DEPARTMENT_ID =
  (SELECT DEPARTMENT_ID FROM HR.DEPARTMENTS WHERE LOCATION_ID = 110);
```

Next, user `tjones` will analyze the `HR.EMPLOYEES` table.

4. Run either of the following scripts to create the `CHAINED_ROWS` table

```
@$ORACLE_HOME/rdbms/admin/utlchain.sql
```

Or

```
@$ORACLE_HOME/rdbms/admin/utlchn1.sql
```

5. Run the `ANALYZE TABLE` statement on the `HR.EMPLOYEES` table.

```
ANALYZE TABLE HR.EMPLOYEES LIST CHAINED ROWS INTO CHAINED_ROWS;
```

Step 4: Disable the Privilege Analysis Policy

You must disable the policy before you can generate a report that captures the actions of user `tjones`.

1. Connect as user `pa_admin`.

```
CONNECT pa_admin -- Or, CONNECT pa_admin@hrpdb
Enter password: password
```

2. Disable the `dba_tuning_priv_analysis_pol` privilege policy.

```
EXEC DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE ('dba_tuning_priv_analysis_pol');
```

Step 5: Generate and View Privilege Analysis Reports

With the privilege analysis policy disabled, user `pa_admin` can generate and view privilege analysis reports.

1. As user `pa_admin`, generate the privilege analysis results.

```
EXEC DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ('dba_tuning_priv_analysis_pol');
```

The generated results are stored in the privilege analysis data dictionary views, which are described in Privilege Analysis Policy and Report Data Dictionary Views.

2. Enter the following commands to format the data dictionary view output:

```
col username format a8
col sys_priv format a18
col used_role format a20
col path format a150
col obj_priv format a10
col object_owner format a10
```

```
col object_name format a10
col object_type format a10
```

3. Find the system privileges and roles that user `tjones` used during the privilege analysis period.

```
SELECT USERNAME, SYS_PRIV, USED_ROLE, PATH
 FROM DBA_USED_SYSPRIVS_PATH
 WHERE USERNAME = 'TJONES'
 ORDER BY 1, 2, 3;
```

Output similar to the following appears:

```
USERNAME SYS_PRIV           USED_ROLE
-------- ------------------ --------------------
PATH
--------------------------------------------------------------------------------
TJONES   ANALYZE ANY        IMP_FULL_DATABASE
GRANT_PATH('TJONES', 'DBA')

TJONES   ANALYZE ANY        IMP_FULL_DATABASE
GRANT_PATH('TJONES', 'DBA', 'IMP_FULL_DATABASE')

TJONES   ANALYZE ANY        IMP_FULL_DATABASE
GRANT_PATH('TJONES', 'DBA', 'DATAPUMP_IMP_FULL_DATABASE', 'IMP_FULL_DATABASE')
...
```

4. Find the object privileges and roles that user `tjones` used during the privilege analysis period.

```
col username format a9
col used_role format a10
col object_name format a22
col object_type format a12

SELECT USERNAME, OBJ_PRIV, USED_ROLE,
 OBJECT_OWNER, OBJECT_NAME, OBJECT_TYPE
 FROM DBA_USED_OBJPRIVS
 WHERE USERNAME = 'TJONES'
 ORDER BY 1, 2, 3, 4, 5, 6;
```

Output similar to the following appears:

```
USERNAME  OBJ_PRIV   USED_ROLE  OBJECT_OWN OBJECT_NAME            OBJECT_TYPE
--------- ---------- ---------- ---------- ---------------------- ------------
TJONES    EXECUTE    PUBLIC     SYS        DBMS_APPLICATION_INFO  PACKAGE
TJONES    SELECT     PUBLIC     SYS        DUAL                   TABLE
TJONES    SELECT     PUBLIC     SYS        DUAL                   TABLE
TJONES    SELECT     PUBLIC     SYSTEM     PRODUCT_PRIVS          VIEW
...
```

5. Find the unused system privileges for user `tjones`.

```
col username format a9
col sys_priv format a35

SELECT USERNAME, SYS_PRIV
 FROM DBA_UNUSED_SYSPRIVS
 WHERE USERNAME = 'TJONES'
 ORDER BY 1, 2;

USERNAME SYS_PRIV
```

```
-------- ----------------------------
TJONES   ADMINISTER ANY SQL TUNING SET
TJONES   ADMINISTER DATABASE TRIGGER
TJONES   ADMINISTER RESOURCE MANAGER
TJONES   ADMINISTER SQL TUNING SET
TJONES   ALTER ANY ASSEMBLY
TJONES   ON COMMIT REFRESH
...
```

Step 6: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. As user `pa_admin`, drop the `dba_tuning_priv_analysis_pol` privilege analysis policy.

   ```
   EXEC DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE ('dba_tuning_priv_analysis_pol');
   ```

   Even though in the next steps you will drop the `pa_admin` user, including any objects created in this user's schema, you must manually drop the `dba_tuning_priv_analysis_pol` privilege analysis policy because this object resides in the `SYS` schema.

2. Connect as the user who created the user accounts. If Oracle Database Vault is enabled, then connect as the Oracle Database Vault Account Manager.

   For example:

   ```
   CONNECT bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
   Enter password: password
   ```

3. Drop the users `pa_admin` and `tjones`.

   ```
   DROP USER pa_admin;
   DROP USER tjones CASCADE;
   ```

Privilege Analysis Policy and Report Data Dictionary Views

Oracle Database provides a set of data dictionary views that provide information about analyzed privileges.

Table 4-1 lists these data dictionary views.

Table 4-1 Data Dictionary Views That Display Privilege Analysis Information

View	Description
`DBA_PRIV_CAPTURES`	Lists information about existing privilege analysis policies
`DBA_USED_PRIVS`	Lists the privileges and capture runs that have been used for reported privilege analysis policies
`DBA_UNUSED_GRANTS`	Lists the privilege grants that have not been used
`DBA_UNUSED_PRIVS`	Lists the privileges and capture runs that have not been used for reported privilege analysis policies

Table 4-1 (Cont.) Data Dictionary Views That Display Privilege Analysis Information

View	Description
DBA_USED_OBJPRIVS	Lists the object privileges and capture runs that have been used for reported privilege analysis policies. It does not include the object grant paths.
DBA_UNUSED_OBJPRIVS	Lists the object privileges and capture runs that have not been used for reported privilege analysis policies. It does not include the object privilege grant paths.
DBA_USED_OBJPRIVS_PATH	Lists the object privileges and capture runs that have been used for reported privilege analysis policies. It includes the object privilege grant paths.
DBA_UNUSED_OBJPRIVS_PATH	Lists the object privileges and capture runs that have not been used for reported privilege analysis policies. It includes the object privilege grant paths.
DBA_USED_SYSPRIVS	Lists the system privileges and capture runs that have been used for reported privilege analysis policies. It does not include the system privilege grant paths.
DBA_UNUSED_SYSPRIVS	Lists the system privileges and capture runs that have not been used for reported privilege analysis policies. It does not include the system privilege grant paths.
DBA_USED_SYSPRIVS_PATH	Lists the system privileges and capture runs that have been used for reported privilege analysis policies. It includes the system privilege grant paths.
DBA_UNUSED_SYSPRIVS_PATH	Lists the system privileges and capture runs that have not been used for reported privilege analysis policies. It includes system privilege grant paths
DBA_USED_PUBPRIVS	Lists all the privileges and capture runs for the PUBLIC role that have been used for reported privilege analysis policies
DBA_USED_USERPRIVS	Lists the user privileges and capture runs that have been used for reported privilege analysis policies. It does not include the user privilege grant paths.
DBA_UNUSED_USERPRIVS	Lists the user privileges and capture runs that have not been used for reported privilege analysis policies. It does not include the user privilege grant paths.
DBA_USED_USERPRIVS_PATH	Lists the user privileges and capture runs that have been used for reported privilege analysis policies. It includes the user privilege grant paths.
DBA_UNUSED_USERPRIVS_PATH	Lists the privileges and capture runs that have not been used for reported privilege analysis policies. It includes the user privilege grant paths.

See Also:

Oracle Database Reference for a detailed description of these data dictionary views

5

Configuring Realms

You can create a realm around database objects to protect them, and then set authorizations to control user access to this data.

- What Are Realms?
  Realms enable you to protect database objects, including specific object types.
- Default Realms
  Oracle Database Vault provides default realms, which are regular realms, not mandatory realms.
- Creating a Realm
  To enable realm protection, you create the realm and configure it to include realm-secured objects, roles, and authorizations.
- About Realm-Secured Objects
  Realm-secured objects define the territory—a set of schema and database objects and roles—that a realm protects.
- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.
- Realm Authorizations in a Multitenant Environment
  In a multitenant environment, the rules and behavior for common realm authorizations are similar to the authorizations for other common objects.
- Modifying the Enablement Status of a Realm
  You can disable or enable a realm, or set the realm to use simulation mode, from Enterprise Manager Cloud Control.
- Deleting a Realm
  You can use Enterprise Manager Cloud Control to delete realms.
- How Realms Work
  When an appropriately privileged database account issues a SQL statement that affects an object within a realm, a special set of activities occur.
- How Authorizations Work in a Realm
  Realm authorizations prevent users from performing activities if the users do not have the correct privileges.
- Access to Objects That Are Protected by a Realm
  You can protect an object by a realm, but still enable access to objects that are part of this realm-protected object.
- Example of How Realms Work
  Realms can provide protection in which two users who each have the same privileges must have separate access levels for an object.
- How Realms Affect Other Oracle Database Vault Components
  Realms have no effect on factors, identities, or rule sets, but they do affect command rules.

- Guidelines for Designing Realms
  Oracle provides a set of guidelines for designing realms.
- How Realms Affect Performance
  Realms can affect database performance in a variety situations, such as with DDL and DML operations.
- Realm Related Reports and Data Dictionary Views
  Oracle Database Vault provides reports and data dictionary views that are useful for analyzing realms.

What Are Realms?

Realms enable you to protect database objects, including specific object types.

- About Realms
  A realm is a grouping of database schemas, database objects, and database roles that must be secured for a given application.
- Mandatory Realms to Restrict User Access to Objects within a Realm
  By default, users who own or have object privileges are allowed to access realm-protected objects without explicit realm authorization.
- Realms in a Multitenant Environment
  In a multitenant environment, you can create a realm to protect common objects in the application root.
- Object Types That Realms Can Protect
  You can create realms around all objects in a schema of certain object types.

About Realms

A realm is a grouping of database schemas, database objects, and database roles that must be secured for a given application.

Think of a realm as zone of protection for your database objects. A schema is a logical collection of database objects such as tables, views, and packages, and a role is a collection of privileges. By arranging schemas and roles into functional groups, you can control the ability of users to use system privileges against these groups and prevent unauthorized data access by the database administrator or other powerful users with system privileges. Oracle Database Vault does not replace the discretionary access control model in the existing Oracle database. It functions as a layer on top of this model for both realms and command rules.

Oracle Database Vault provides two types of realms: regular and mandatory. Both realm types can protect either an entire schema or crucial objects within a schema selectively, such as tables and indexes. With a regular realm, an object owner or users who has been granted object privileges can perform queries or DML operations without realm authorization but must have realm authorization to perform DDL operations. A mandatory realm provides stronger protection for objects within a realm. Mandatory realms block both object privilege-based and system privilege-based access and will not allow users with object privileges to perform queries, DML, or DDL operations without realm authorization. In other words, even an object owner cannot access his or her own objects without proper realm authorization if the objects are protected by mandatory realms.

For databases that use Oracle Flashback Technology, then both regular and mandatory realms will enforce the same behavior for a flashback table. Users can

execute a `FLASHBACK TABLE` SQL statement on a realm-protected table if the user is authorized to the realm.

For databases that use Information Lifecycle Management (ILM), a Database Vault administrator can use the `DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER` and `DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER` procedure to control who can perform ILM operations on realm-protected objects.

You can register schemas, all objects of a certain type in a schema, or individual objects within a schema into a realm. After you create a realm, you can register a set of schema objects or roles (secured objects) for realm protection and authorize a set of users or roles to access the secured objects. Objects that are protected by a regular realm allow DML access to users who have direct object grants.

For example, you can create a realm to protect all existing database schemas that are used in an accounting department. The realm prohibits any user who is not authorized to the realm to use system privileges to access the secured accounting data. When an entire schema is protected, all objects in the schema are protected, including tables, indexes, procedures and other objects.

You can run reports on realms that you create in Oracle Database Vault. You can use simulation mode during development, test, and even production phases to log only realm violations instead of blocking access. This enables you to quickly test applications using Database Vault realms.

You can configure realms by using the Oracle Database Vault Administrator pages in Oracle Enterprise Manager Cloud Control. Alternatively, you can configure realms by using the PL/SQL interfaces and packages provided by Oracle Database Vault.

Related Topics

- Oracle Database Vault Realm APIs
  The `DBMS_MACADM` PL/SQL package enables you to configure Oracle Database Vault realms.

Mandatory Realms to Restrict User Access to Objects within a Realm

By default, users who own or have object privileges are allowed to access realm-protected objects without explicit realm authorization.

You optionally can configure the realm to prevent these users' access by configuring it to be a mandatory realm. Mandatory realms block system privilege-based access as well as object privilege-based access. This means that even the object owner cannot have access if he or she is not authorized to access the realm. Users can access secured objects in the mandatory realm only if the user or role is authorized to do so.

Mandatory realms have the following additional characteristics:

- If there are multiple mandatory realms on the same object, then you must authorize the user or role on all the mandatory realms before they can access the protected object.
- If a role is protected by a mandatory realm, then no privileges can be granted to or revoked from the protected role except by the realm owner.
- You can update regular realms that you created in earlier releases to be mandatory realms. This way, you can block owner access and object-privileged users from accessing the realm-protected objects.

- SYS-owned objects are already protected by data dictionary protection and are not protected separately by Oracle Database Vault.

Mandatory realms have the following benefits:

- **Mandatory realms can block object owners and object privileged users.** In previous releases, blocking these users could only be done by defining complicated command rules.
- **Mandatory realms provide more flexible configurations for access control.** For example, suppose you want to enable a user to access an object with certain conditions, such as in a specific time range during the day. You cannot grant object privileges to that user because realms do not block object privileges. You only can grant system privileges to the user and then authorize this user to the realm with a rule, or make a command rule on the command directly. These solutions are either very expensive in terms of computational cost or undesirable because they entail the excessive granting of privileges such as system privileges to the user. With a mandatory realm, you only need to grant object privileges to the user, with a rule for specific conditions, and then authorize this user to be a realm owner or participant. Thus, with mandatory realms, Oracle Database Vault policies have more flexibility without granting users excessive privileges.
- **Mandatory realms add a layer of protection during patch upgrades.** During a patch upgrade, a database administrator may need to have direct access to a realm-protected object in order to perform a patch on the object. If there are tables that contain sensitive data, such as social security numbers, you can protect these tables from the administrator's access with mandatory realms during the patch upgrade. When patching is complete, and the database administrator no long needs access to the objects, you can disable mandatory realm protection and then re-enable the normal application realm protection so that the application protection can return to its normal state.
- **You can use mandatory realms to secure tables during runtime.** During runtime, application data can be stored in many tables. It is better to have a single user such as a runtime schema to access these tables so that you can maintain the integrity and correctness of the data. If the application data is scattered in many different schemas, then schema owners and users with object privileges can change the data if they log into the database directly. To insure that users cannot update these tables without going through the runtime schema's procedures, you can use mandatory realms to protect the tables so that only the authorized user's procedures can access them. Because a regular realm does not block object owners and object-privileged users, you can use mandatory realms to block them. This way, only authorized users can access these tables during runtime.
- **You can freeze security settings by preventing changes to configured roles.**

Related Topics

- CREATE_REALM Procedure
  The `CREATE_REALM` procedure creates a realm. In a multitenant environment, you can create both common and local realms.
- UPDATE_REALM Procedure
  The `UPDATE_REALM` procedure updates a realm.

Realms in a Multitenant Environment

In a multitenant environment, you can create a realm to protect common objects in the application root.

The advantage of creating a realm in the application root instead of creating a large number objects and realms around these objects within individual pluggable databases (PDBs) is that you can create them in one place, the application root. This way, you can manage them centrally.

You cannot create a common realm in the CDB root.

A Database Vault common realm can be either a regular realm or a mandatory realm. The realm protects only objects within the application root, not local objects in a PDB. The CDB root, application root, and any affected PDBs all must be Database Vault enabled.

To configure a common realm, you must be commonly granted the `DV_OWNER` or `DV_ADMIN` role. To grant common authorizations for a common realm, you must be in the application root. To propagate the realm to the PDBs that are associated with the application root, you must synchronize the application root. For example, to synchronize an application called `saas_sales_app`

```
ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;
```

Related Topics

- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.

Object Types That Realms Can Protect

You can create realms around all objects in a schema of certain object types.

These object types are as follows:

Object Types C-J	Object Types L-P	Object Types R-V
CLUSTER	LIBRARY	ROLE
DIMENSION	MATERIALIZED VIEW	SEQUENCE
FUNCTION	MATERIALIZED VIEW LOG	SYNONYM
INDEX	OPERATOR	TABLE
INDEX PARTITION	PACKAGE	TRIGGER
INDEXTYPE	PROCEDURE	TYPE
JOB	PROGRAM	VIEW

Default Realms

Oracle Database Vault provides default realms, which are regular realms, not mandatory realms.

- Oracle Database Vault Realm
  The Oracle Database Vault realm protects configuration and role information in the Oracle Database Vault `DVSYS`, `DVF`, and `LBACSYS` schemas.

- Database Vault Account Management Realm
  The Database Vault Account Management realm defines the realm for the administrators who manage and create database accounts and database profiles.
- Oracle Enterprise Manager Realm
  Oracle Database Vault provides a realm specifically for Oracle Enterprise Manager accounts.
- Oracle Default Schema Protection Realm
  Oracle Default Schema Protection Realm protects roles and schemas that are used with Oracle features such as Oracle Text.
- Oracle System Privilege and Role Management Realm
  Oracle System Privilege and Role Management Realm protects sensitive roles that are used to export and import data in an Oracle database.
- Oracle Default Component Protection Realm
  Oracle Default Component Protection Realm protects the `SYSTEM` and `OUTLN` schemas.

Oracle Database Vault Realm

The Oracle Database Vault realm protects configuration and role information in the Oracle Database Vault `DVSYS`, `DVF`, and `LBACSYS` schemas.

The owners of all three of the `DVSYS`, `DVF`, and `LBACSYS` schemas are owners of this realm.

This realm protects the following objects:

- Entire schemas that are protected: `DVSYS`, `DVF`, `LABACSYS`
- Roles that are protected:

Schemas DV_A-DV_S	Schemas DV_P-L	Schemas DV_G-DV_X
DV_ADMIN	DV_MONITOR	DV_GOLDENGATE_ADMIN
DV_AUDIT_CLEANUP	DV_PATCH_ADMIN	DV_GOLDENGATE_REDO_ACCESS
DV_DATAPUMP_NETWORK_LINK	DV_PUBLIC	DV_XSTREAM_ADMIN
DV_OWNER	DV_STREAMS_ADMIN	-
DV_SECANALYST	LBAC_DBA	-

- PL/SQL package that is protected: `SYS.DBMS_RLS`

Related Topics

- Oracle Database Vault Schemas
  The Oracle Database Vault schemas, `DVSYS` and `DVF`, support the administration and run-time processing of Oracle Database Vault.

Database Vault Account Management Realm

The Database Vault Account Management realm defines the realm for the administrators who manage and create database accounts and database profiles.

This realm protects the DV_ACCTMGR and CONNECT roles. The owner of this realm can grant or revoke the CREATE SESSION privilege to or from a user.

Related Topics

- DV_ACCTMGR Database Vault Account Manager Role
  The DV_ACCTMGR role is a powerful role, used for accounts management.

Oracle Enterprise Manager Realm

Oracle Database Vault provides a realm specifically for Oracle Enterprise Manager accounts.

The Oracle Enterprise Manager realm protects Oracle Enterprise Manager accounts that are used for monitoring and management (DBSNMP user and the OEM_MONITOR role).

Oracle Default Schema Protection Realm

Oracle Default Schema Protection Realm protects roles and schemas that are used with Oracle features such as Oracle Text.

The advantage of this grouping is that Oracle Spatial schemas (MDSYS, MDDATA) are used extensively with Oracle Text (CTXSYS), and Oracle OLAP is an application rather than a core Oracle Database kernel feature.

Oracle Default Schema Protection Realm Protected Roles and Schemas

Oracle Default Schema Protection Realm protects several roles and schemas.

- Roles that are protected by default: CTXAPP, OLAP_DBA, EJBCLIENT, OLAP_USER
- Schemas that are protected by default: CTXSYS, EXFSYS, MDDATA, MDSYS
- Roles that are recommended for protection: APEX_ADMINISTRATOR_ROLE, SPATIAL_CSW_ADMIN, WFS_USR_ROLE, CSW_USR_ROLE, SPATIAL_WFS_ADMIN, WM_ADMIN_ROLE
- Schemas that are recommended for protection: APEX_030200, OWBSYS, WMSYS

Oracle Default Schema Protection Realm Owners

The SYS, CTXSYS, and EXFSYS users are the default owners of Oracle Default Schema Protection Realm. These users can grant the roles protected by this realm to other users, and grant permissions on its schemas to other users as well.

Oracle System Privilege and Role Management Realm

Oracle System Privilege and Role Management Realm protects sensitive roles that are used to export and import data in an Oracle database.

This realm also contains authorizations for users who must grant system privileges.

User SYS is the only default owner of this realm. Any user who is responsible for managing system privileges should be authorized as an owner to this realm. These users can grant the roles that are protected by this realm to other users.

- Roles that are protected by default:

Roles A-E	Roles G-J	Roles J-S
AQ_ADMINISTRATOR_ROLE	GATHER_SYSTEM_STATISTICS	JAVAUSERPRIV
AQ_USER_ROLE	GLOBAL_AQ_USER_ROLE	LOGSTDBY_ADMINISTRATOR
DBA	HS_ADMIN_ROLE	OPTIMIZER_PROCESSING_RATE
DBA_OLS_STATUS	IMP_FULL_DATABASE	RECOVERY_CATALOG_OWNER
DELETE_CATALOG_ROLE	JAVA_ADMIN	RESOURCE
DV_REALM_OWNER	JAVADEBUGPRIV	SCHEDULER_ADMIN
DV_REALM_RESOURCE	JAVA_DEPLOY	SELECT_CATALOG_ROLE
EXECUTE_CATALOG_ROLE	JAVAIDPRIV	-
EXP_FULL_DATABASE	JAVASYSPRIV	-

- Roles that are recommended for protection: `DBFS_ROLE`, `HS_ADMIN_EXECUTE_ROLE`, `HS_ADMIN_SELECT_ROLE`

Oracle Default Component Protection Realm

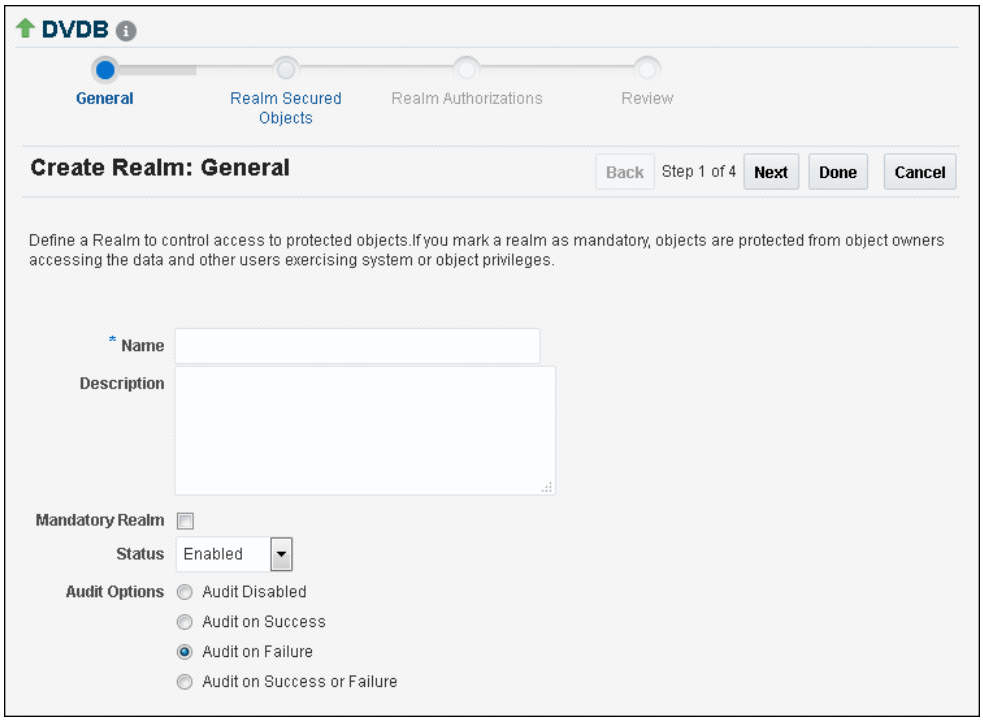
Oracle Default Component Protection Realm protects the `SYSTEM` and `OUTLN` schemas.

The authorized users of this realm are users `SYS` and `SYSTEM`.

Creating a Realm

To enable realm protection, you create the realm and configure it to include realm-secured objects, roles, and authorizations.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Realms**.
3. In the Realms page, click **Create** to display the Create Realm page.



4. In the Create Realm page, enter the following settings:
   - **Name:** Enter a name for the realm. It can contain up to 90 characters in mixed-case. This attribute is mandatory.

     Oracle suggests that you use the name of the protected application as the realm name (for example, `hr_app` for an human resources application).
   - **Description:** Enter a brief description of the realm. The description can contain up to 1024 characters in mixed-case. This attribute is optional.

     You may want to include a description for the business objective of the given application protection and document all other security policies that compliment the realm's protection. Also document who is authorized to the realm, for what purpose, and any possible emergency authorizations.
   - **Mandatory Realm:** Select this check box to create the realm as a mandatory realm. See Mandatory Realms to Restrict User Access to Objects within a Realm for more information about mandatory realms.
   - **Status:** Select either **Enabled**, **Disabled**, or **Simulation**. This attribute is mandatory.
   - **Audit Options:** Select one of the following:
     - **Audit Disabled:** Does not create an audit record.
     - **Audit on Success:** Creates an audit record for authorized activities.
     - **Audit on Failure:** Creates an audit record when a realm violation occurs (for example, when an unauthorized user tries to modify an object that is protected by the realm).
     - **Audit on Success or Failure:** Creates an audit record for any activity that occurs in the realm, including both authorized and unauthorized activities.

In a non-unified auditing environment, Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` table. See Auditing Oracle Database Vault for more information. If you have enabled unified auditing, then this setting does not capture audit records. Instead, you must create audit policies to capture this information, as described in *Oracle Database Security Guide*. Oracle Database also provides a default policy, `ORA_DV_AUDPOL`, that audits all actions that are performed on the Oracle Database Vault `DVSYS` and `DVF` schema objects and the Oracle Label Security `LBACSYS` schema objects.

5. Click **Next** to display the Realm secured objects page.

   See About Realm-Secured Objects for conceptual information about the settings for this page.

6. Click the **Add** button, and in the Add Secured Object dialog box, enter the following information:

   - **Object Owner:** From the list, select the name of the database schema owner. You can enter the `%` character if the object you want to secure with the realm is a role. This attribute is mandatory.
   - **Object Type:** From the list, select the type of the database object, such as `TABLE`, `INDEX`, or `ROLE`. This attribute is mandatory.

     You can add as many objects of any type as you want to the realm.

     By default, the **Object Type** box contains the % wildcard character to include all object types for the specified **Object Owner**. However, it does not include roles, which do not have specific schema owners in the database and must be specified explicitly.
   - **Object Name:** Enter the name of the object in the database that the realm must protect, or enter `%` to specify all objects (except roles) for the object owner that you have specified. If you enter `%`, then it can encompass all objects in the schema if `%` is also used for the **Object Type** setting. But if **Object Type** is set to **Tables**, then using `%` for the **Object Name** refers to all tables in the schema. This attribute is mandatory.

     By default, the **Object Name** field contains the % wildcard character to encompass the entire schema specified for **Object Type** and **Object Owner**. Note that the % wildcard character applies to objects that do not yet exist and currently existing objects.

     If you enter `%` in the **Object Name** field, then it encompasses all objects in the schema if you also enter `%` in the **Object Type** field. However, if you set **Object Type** to a specific object type, such as **Tables**, then the setting of `%` in **Object Type** refers to all objects of that type (in this case, tables) in the schema.

7. Click **Next** to display the Realm authorizations page.

   See About Realm Authorization for conceptual information about the settings for this page.

8. Click the **Add** button, and in the Add Authorizations dialog box, enter the following information:

   - **Realm Authorization Grantee:** From the list, select the database account or role to whom you want to grant the realm authorization.

     This list shows all accounts and roles in the system, not just accounts with system privileges.

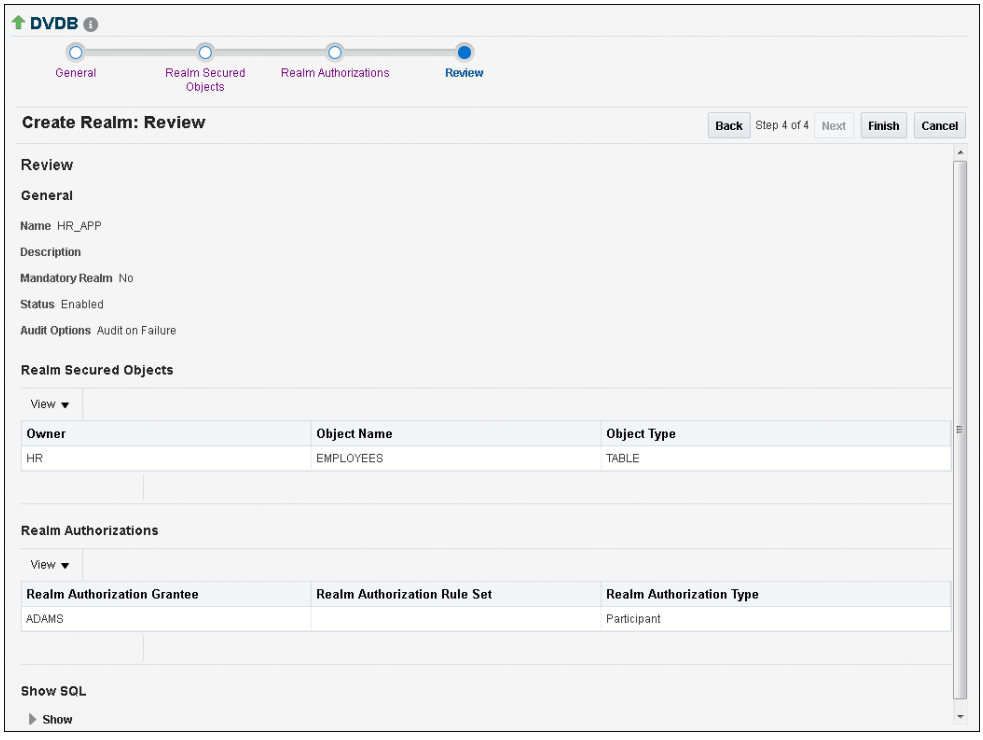
- **Realm Authorization Type:** Select either of the following settings. This attribute is mandatory.
  - **Participant:** This account or role can exercise system privileges to access, manipulate, and create objects protected by the realm, provided that these privileges have been granted using the standard Oracle Database privilege grant process. A realm can have multiple participants.
  - **Owner:** This account or role has the same rights as the realm participant, plus the authorization to grant or revoke realm-secured database roles. The realm owner can grant or revoke privileges on realm-protected objects to other users. A realm can have multiple owners.
- **Realm Authorization Rule Set:** Select from the available rule sets that have been created for your site. You can select only one rule set, but the rule set can have multiple rules.

  See Creating a Rule to Add to a Rule Set for more information about defining rules to govern the realm authorization.

  Any auditing and custom event handling associated with the rule set occurs as part of the realm authorization processing.

9. Click **Next** to display the Review page.
10. In the Review page, check the settings you have created.

    For example:



11. Click **Finish** to complete the realm creation.

Related Topics

- About Realm-Secured Objects
  Realm-secured objects define the territory—a set of schema and database objects and roles—that a realm protects.
- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.
- Propagating Oracle Database Vault Configurations to Other Databases
  You can propagate Database Vault configurations (such as a realm configuration) to other Database Vault-protected databases.

About Realm-Secured Objects

Realm-secured objects define the territory—a set of schema and database objects and roles—that a realm protects.

You can create the following types of protections:

- Objects from multiple database accounts or schemas can be under the same realm.
- One object can belong to multiple realms.

  If an object belongs to multiple realms, then Oracle Database Vault checks the realms for the proper authorization. For `SELECT`, DDL, and DML statements, as long as a user is a participant in one of the realms, and if the command rules permit it, then the commands that the user enters are allowed. For `GRANT` and `REVOKE` operations of a database role in multiple realms, the person performing the `GRANT` or `REVOKE` operation must be the realm owner. Schema owners can perform DML operations on objects that are protected by multiple regular realms.

  If one of the realms is a mandatory realm, then the user who wants to access the object must be a realm owner or participant in the mandatory realm. During the authorization checking process, the non-mandatory realms are ignored. If there are multiple mandatory realms that protect the object, then the user who wants to access the object must be authorized in all of the mandatory realms.
- `SYS`-owned objects are already protected by data dictionary protection and are not protected separately by Oracle Database Vault.

About Realm Authorization

Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.

You can grant a realm authorization to an account or role to allow the use of its system privileges in the following situations:

- When the user must create or access realm-secured objects
- When a user must grant or revoke realm-secured roles

A user who has been granted realm authorization as either a realm owner or a realm participant can use its system privileges to access secured objects in the realm.

Note the following:

- Realm owners cannot add other users to their realms as owners or participants. Only users who have the DV_OWNER or DV_ADMIN role are allowed to add users as owners or participants to a realm.
- Users who have been granted the DV_OWNER role can add themselves to a realm authorization.
- A realm owner, but not a realm participant, can grant or revoke realm secured roles or grant or revoke object privileges on realm secured objects to anyone.
- A user can be granted either as a realm owner or a realm participant, but not both. However, you can update the authorization types of existing realm authorizations.

Use the Edit Realm page to manage realm authorizations. You can create, edit, and remove realm authorizations.

Related Topics

- Realm Authorization Configuration Issues Report
  The Realm Authorization Configuration Issues Report displays Oracle Database Vault realm configuration issues.

Realm Authorizations in a Multitenant Environment

In a multitenant environment, the rules and behavior for common realm authorizations are similar to the authorizations for other common objects.

Local Authorization for a Common Realm

The local authorization for a common realm refers to the authorization a user has for the PDB that this user is accessing.

The rules for the local authorization for a common realm are as follows:

- A user who has been commonly granted the DV_OWNER or DV_ADMIN role can grant local authorization to common users, common roles, local users, and local roles. The common DV_OWNER or DV_ADMIN user can also remove local authorization from a common realm in a PDB.
- A local Database Vault administrator can authorize locally (that is, grant local authorizations to both local and common users) within the PDB. A common Database Vault administrator can also grant authorizations in each PDB. A common realm authorization can only be granted by a common Database Vault administrator in the application root.
- The common Database Vault administrator can both add or remove local authorization to and from a common realm from within the PDB.
- If a common user has only local authorization for a common realm, then this user cannot access the common realm in any other PDB than this local authorization.
- A common user or a common role can have both the local authorization and the common authorization to a common realm at the same time. Removing a common user's local authorization from a common realm does not affect the common user's common authorization. Removing a common user's common authorization from a common realm does not affect the common user's local authorization.

Common Authorization for a Common Realm

The common authorization for a common realm refers to the authorization a common user or a common role has in the application root while the authorization takes effect in every container that is Database Vault enabled.

The rules for the local authorization for a common realm are as follows:

- A user who has been commonly granted the `DV_OWNER` or `DV_ADMIN` role can grant common realm authorization to common users or roles in the application root. This common Database Vault administrator can perform the removal of common authorizations while in the application root.
- This common authorization applies to the containers that have been Database Vault enabled in the CDB.
- If a common user is authorized to a common realm in the application root, then this user has access to the objects protected by the common realm in the application root and any application PDBs.
- Any rule sets that are associated with a common realm must be common rule sets. The rules that are added to a common rule set that is associated with common authorization cannot involve any local objects.

How the Authorization of a Realm Works in Both the Application Root and in an Individual PDB

During the Database Vault enforcement in a container, a common realm performs the same enforcement behaviors as the same realm when it is used locally in a PDB.

Modifying the Enablement Status of a Realm

You can disable or enable a realm, or set the realm to use simulation mode, from Enterprise Manager Cloud Control.

If the realm is managed by a policy, and if the policy status is set to partial, then you can modify the enablement status of the realm. If the policy is set to enabled, disabled, or simulation mode, then you cannot modify the enablement status of the realm.

1. In the Oracle Database Vault Administration page, select **Realms**.
2. In the Realms page, select the realm you want to disable or enable, and then select **Edit**.
3. In the Edit Realm page, under Status in the General section, select either **Disabled**, **Enabled**, or **Simulation**.
4. Click **Done**, and then click **Finished**.

Deleting a Realm

You can use Enterprise Manager Cloud Control to delete realms.

1. Locate the various references to the realm that you want to delete by querying the realm-related Oracle Database Vault data dictionary views.
2. If the Realm is part of a policy, then remove the realm from the policy.
   a. In the Oracle Database Vault Administration page, select **Policies**.
   b. Select the policy that contains the realm and then click **Edit**.
   c. Expand the Realms area.

d. Select the realm and then click **Remove**.

e. Click **Next**, then **Finish**.

3. In the Administration page, under Database Vault Components, select **Realms**.
4. In the Realms page, select the realm you want to delete, and then select **Remove**.
5. In the Confirmation window, click **Yes**.

   Oracle Database Vault deletes the configuration for the realm, including realm authorizations. It does not delete the rule sets used for realm authorizations.

Related Topics

- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

How Realms Work

When an appropriately privileged database account issues a SQL statement that affects an object within a realm, a special set of activities occur.

These privileges include DDL, DML, `EXECUTE`, `GRANT`, `REVOKE`, or `SELECT` privileges.

1. Does the SQL statement affect objects secured by a realm?

   If yes, then go to Step 2. If no, then realms do not affect the SQL statement. Go to Step 7. If the object affected by the command is not secured in any realms, then realms do not affect the SQL statement being attempted.

2. Is the realm a mandatory realm or regular realm?

   If yes, then go to Step 4. If it is regular realm, then go to Step 3.

3. Is the database account using a system privilege to execute the SQL statement?

   If yes, then go to Step 4. If no, then go to Step 6. If the session has object privileges on the object in question for `SELECT`, `EXECUTE`, and DML statements only, then the realm protection is not enforced. Realms protect against the use of any system privilege on objects or roles protected by the realm. Even users with object privileges for objects that are protected by regular realms are prevented from performing DDL operations.

   Remember that if the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter has been set to `TRUE`, then non-`SYS` users have access to `SYS` schema objects. For better security, ensure that `O7_DICTIONARY_ACCESSIBILITY` is set to `FALSE`.

4. Is the database account a realm owner or realm participant?

   If yes, then go to Step 5. Otherwise, a realm violation occurs and the statement is not allowed to succeed. If the command is a `GRANT` or `REVOKE` of a role that is protected by the realm, or the `GRANT` or `REVOKE` of an object privilege on an object protected by the realm, then the session must be authorized as the realm owner directly or indirectly through roles.

5. Is the realm authorization for the database account conditionally based on a rule set?

   If yes, then go to Step 6. If no, then go to Step 7.

6. Does the rule set evaluate to `TRUE`?

If yes, then go to Step 7. If no, then there is a realm violation, so the SQL statement is not allowed to succeed.

7. Does a command rule prevent the command from executing?

   If yes, then there is a command rule violation and the SQL statement fails. If no, then there is no realm or command rule violation, so the command succeeds.

   For example, the `HR` account may have the `DROP ANY TABLE` privilege and may be the owner of the `HR` realm, but a command rule can prevent `HR` from dropping any tables in the `HR` schema unless it is during its monthly maintenance window. Command rules apply to the use of the `ANY` system privileges and object privileges and are evaluated after the realm checks.

In addition, because a session is authorized in a realm, it does not mean the account has full control on objects protected by the realm. Realm authorization does *not* implicitly grant extra privileges to the account. The account still must have system privileges or object privileges to access the objects. For example, an account or role may have the `SELECT ANY` table privilege and be a participant in the `HR` realm. This means the account or the account granted the role could query the `HR.EMPLOYEES` table. Being a participant in the realm does not mean the account or role can `DROP` the `HR.EMPLOYEES` table. Oracle Database Vault does not replace the discretionary access control model in the existing Oracle database. It functions as a layer on top of this model for both realms and command rules.

Note the following:

- Protecting a table in a realm does not protect the view by default. Any view that must be protected should be added to the realm regardless of whether the view was created before or after the table was added to the realm.
- For invoker's right procedures that access realm protected objects, the invoker of the procedure must be authorized to the realm.
- Be aware that realm protection does not protect a table if access to the table has been granted to `PUBLIC`. For example, if `SELECT ON` *`table_name`* is granted to `PUBLIC`, then every user has access to *`table_name`* (unless the table is protected by a mandatory realm), even if this table is protected by a realm. As a best practice, revoke unnecessary privileges from `PUBLIC`.

How Authorizations Work in a Realm

Realm authorizations prevent users from performing activities if the users do not have the correct privileges.

- About Authorizations in a Realm
  Realms protect data from access through system privileges.
- Examples of Realm Authorizations
  You can create realms that protect objects from users who have system privileges and other powerful privileges, for example.

About Authorizations in a Realm

Realms protect data from access through system privileges.

Realms do not give additional privileges to the data owner or participants.

The realm authorization provides a run-time mechanism to check logically if a user's command should be allowed or denied to access objects specified in the command and to proceed with its execution.

System privileges are sweeping database privileges such as `CREATE ANY TABLE` and `DELETE ANY TABLE`. These privileges typically apply across schemas and bypass the need for object privileges. Data dictionary views such as `DBA_SYS_PRIVS`, `USER_SYS_PRIVS`, and `ROLE_SYS_PRIVS` list the system privileges for database accounts and roles. Database authorizations work normally for objects not protected by a realm. However, a user must be authorized as a realm owner or participant to successfully use his or her system privileges on objects secured by the realm. A realm violation prevents the use of system privileges and can be audited.

Mandatory realms block both object privileged-based access and system privilege-based access. This means that even the object owner cannot have access if he or she is not authorized to access the realm. Users can access secured objects in the mandatory realm only if the user or role is authorized to do so.

Examples of Realm Authorizations

You can create realms that protect objects from users who have system privileges and other powerful privileges, for example.

- Example: Unauthorized User Trying to Create a Table
  The `ORA-47401` error appears when unauthorized users try to create tables.
- Example: Unauthorized User Trying to Use the DELETE ANY TABLE Privilege
  An `ORA-01031: insufficient privileges` error appears for unauthorized user access.
- Example: Authorized User Performing DELETE Operation
  Authorized users are allowed to perform the activities for which they are authorized.

Example: Unauthorized User Trying to Create a Table

The `ORA-47401` error appears when unauthorized users try to create tables.

Example 5-1 shows what happens when an unauthorized user who has the `CREATE ANY TABLE` system privilege tries to create a table in a realm where the `HR` schema is protected by a realm.

Example 5-1 Unauthorized User Trying to Create a Table

```
CREATE TABLE HR.demo2 (col1 NUMBER(1));
```

The following output should appear

```
ORA-47401: Realm violation for CREATE TABLE on HR.DEMO2
```

As you can see, the attempt by the unauthorized user fails. Unauthorized use of system privileges such as `SELECT ANY TABLE`, `CREATE ANY TABLE`, `DELETE ANY TABLE`, `UPDATE ANY TABLE`, `INSERT ANY TABLE`, `CREATE ANY INDEX`, and others results in failure.

Example: Unauthorized User Trying to Use the DELETE ANY TABLE Privilege

An `ORA-01031: insufficient privileges` error appears for unauthorized user access.

Example 5-2 shows what happens when an unauthorized database account tries to use his `DELETE ANY TABLE` system privilege to delete an existing record, the database session returns the following error.

Example 5-2 Unauthorized User Trying to Use the DELETE ANY TABLE Privilege

```
DELETE FROM HR.EMPLOYEES WHERE EMPNO = 8002;
```

The following output should appear:

```
ERROR at line 1:
ORA-01031: insufficient privileges
```

Realms do not affect direct privileges on objects. For example, a user granted delete privileges to the `HR.EMPLOYEES` table can successfully delete records without requiring realm authorizations. Therefore, realms should minimally affect normal business application usage for database accounts.

Example: Authorized User Performing DELETE Operation

Authorized users are allowed to perform the activities for which they are authorized.

Example 5-3 shows how an authorized user can perform standard tasks allowed within the realm.

Example 5-3 Authorized User Performing DELETE Operation

```
DELETE FROM HR.EMPLOYEES WHERE EMPNO = 8002;

1 row deleted.
```

Access to Objects That Are Protected by a Realm

You can protect an object by a realm, but still enable access to objects that are part of this realm-protected object.

For example, suppose you create a realm around a specific table. However, you want users to be able to create an index on this table. You can accomplish this as follows, depending on the following scenarios.

- **The user does not have the CREATE ANY INDEX privilege.** As the realm owner of the table, grant the `CREATE INDEX ON` *`table`* privilege to the user who must create the index.
- **The user has the CREATE ANY INDEX privilege.** In this case, create another realm and make all index types as the secured objects and grant that user participant authorization to the realm. (Remember that having the `CREATE ANY INDEX` privilege alone is not sufficient for a non-realm participant to create an index in a realm-protected table.)

- **You want all of your database administrators to be able to create an index and they have the CREATE ANY INDEX privilege.** In your data protection realm, specify all object types to be protected *except* the index types. This permits all of your administrators to create indexes for the protected table.

Example of How Realms Work

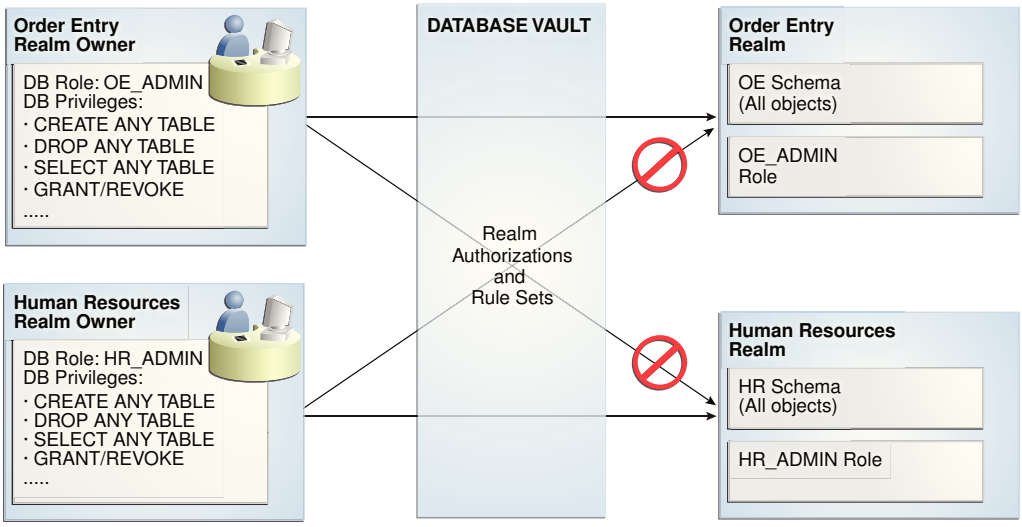
Realms can provide protection in which two users who each have the same privileges must have separate access levels for an object.

Figure 5-1 illustrates how data within a realm is protected.

In this scenario, two users, each in charge of a different realm, have the same system privileges. The owner of a realm can be either a database account or a database role. As such, each of the two roles, `OE_ADMIN` and `HR_ADMIN`, can be protected by a realm as a secured object *and* be configured as the owner of a realm.

Further, only a realm owner, such as `OE_ADMIN`, can grant or revoke database roles that are protected by the realm. The realm owner cannot manage roles protected by other realms such as the `DBA` role created by `SYS` in the Oracle System Privilege and Role Management realm. Any unauthorized attempt to use a system privilege to access realm-protected objects raises a realm violation, which can be audited. The powers of each realm owner are limited within the realm itself. For example, `OE_ADMIN` has no access to the Human Resources realm, and `HR_ADMIN` has no access to the Order Entry realm.

Figure 5-1 How Authorizations Work for Realms and Realm Owners



Related Topics

- Quick Start Tutorial: Securing a Schema from DBA Access
  This tutorial shows how to create a realm around the `HR` schema.

How Realms Affect Other Oracle Database Vault Components

Realms have no effect on factors, identities, or rule sets, but they do affect command rules.

With command rules, Oracle Database Vault evaluates the realm authorization first when processing SQL statements.

How Realms Work explains the steps that Oracle Database Vault takes to process SQL statements that affect objects in a realm. How Command Rules Work describes how command rules are processed.

Guidelines for Designing Realms

Oracle provides a set of guidelines for designing realms.

- Create realms based on the schemas and roles that form a database application.

  Define database roles with the minimum and specific roles and system privileges required to maintain the application objects and grant the role to named accounts. You then can add the role as an authorized member of the realm. For object-level privileges on objects protected by the realm and required by an application, create a role and grant these minimum and specific object-level privileges to the role, and then grant named accounts this role. In most cases, these types of roles do not need to be authorized in the realm unless `ANY`-style system privileges are already in use. A model using the principle of least privilege is ideal for any database application.
- A database object can belong to multiple realms and an account or role can be authorized in multiple realms.

  To provide limited access to a subset of a database schema (for example, just the `EMPLOYEES` table in the `HR` schema), or roles protected by a realm, create a new realm with just the minimum required objects and authorizations.
- If you want to add a role to a realm as a grantee, create a realm to protect the role. Doing so prevents users who have been granted the `GRANT ANY ROLE` system privilege, such as the `SYSTEM` user account, from granting the role to themselves.
- If you want to add the `SYS` user account to a realm authorization, you must add user `SYS` explicitly and not through a role (such as the `DBA` role).
- Be mindful of the privileges currently allowed to a role that you plan to add as a realm authorization.

  Realm authorization of a role can be accidentally granted and not readily apparent if an account such as `SYS` or `SYSTEM` creates a role for the first time and the Oracle Database Vault administrator adds this role as a realm authorization. This is because the account that creates a role is implicitly granted the role when it is created.
- Sometimes you must temporarily relax realm protections for an administrative task. Rather than disabling the realm, have the Security Manager (`DV_ADMIN` or `DV_OWNER`) log in, add the named account to the authorized accounts for the realm, and set the authorization rule set to Enabled. Then in the enabled rule set, turn on

all auditing for the rule set. You can remove the realm authorization when the administrative task is complete.

- If you want to grant ANY privileges to new users, Oracle recommends that you add a database administrative user to the Oracle System Privilege and Role Management realm so that this user can grant other users ANY privileges, if they need them. For example, using a named account to perform the GRANT of the ANY operations enables you to audit these operations, which creates an audit trail for accountability.
- If you drop a table, index, or role that has been protected by a realm and then recreate it using the same name, the realm protection is not restored. You must re-create the realm protection for the new table, index, or role. However, you can automatically enforce protection for all future tables, indexes, and roles within a specified schema. For example, to enforce protection for all future tables:

```
BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM('realm_name', 'schema_name', '%', 'TABLE');
END;
/
```

- You can test the development phase of a realm by using simulation mode, which enables the realm but writes detailed information about violations to a log file.

Related Topics

- Using Simulation Mode for Logging Realm and Command Rule Activities
  Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

How Realms Affect Performance

Realms can affect database performance in a variety situations, such as with DDL and DML operations.

- **DDL and DML operations on realm-protected objects do not have a measurable effect on Oracle Database.** Oracle recommends that you create the realm around the entire schema, and then authorize specific users to perform only specific operations related to their assigned tasks. For finer-grained control, you can define realms around individual tables and authorize users to perform certain operations on them, and also have a realm around the entire schema to protect the entire application. Be aware, however, that this type of configuration may slow performance, but it does enable you to grant realm authorization to some of the objects in a schema.
- **Auditing affects performance.** To achieve the best performance, Oracle recommends that you use fine-grained auditing rather than auditing all operations.
- **Periodically check the system performance.** You can do so by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Cloud Control, which is installed by default with Oracle Database), Automatic Workload Repository (AWR), and TKPROF.

See Also:

- *Oracle Database Performance Tuning Guide* to learn how to monitor database performance
- *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements

Realm Related Reports and Data Dictionary Views

Oracle Database Vault provides reports and data dictionary views that are useful for analyzing realms.

Table 5-1 lists the Oracle Database Vault reports. See Oracle Database Vault Reports, for information about how to run these reports.

Table 5-1 Reports Related to Realms

Report	Purpose
Realm Audit Report	Audits records generated by the realm protection and realm authorization operations
Realm Authorization Configuration Issues Report	Lists authorization configuration information, such as incomplete or disabled rule sets, or nonexistent grantees or owners that may affect the realm
Rule Set Configuration Issues Report	Lists rule sets that do not have rules defined or enabled, which may affect the realms that use them
Object Privilege Reports	Lists object privileges that the realm affects
Privilege Management - Summary Reports	Provides information about grantees and owners for a realm
Sensitive Objects Reports	Lists objects that the command rule affects

Table 5-2 lists data dictionary views that provide information about existing realms.

Table 5-2 Data Dictionary Views Used for Realms

Data Dictionary View	Description
DBA_DV_REALM View	Lists the realms created in the current database instance.
DBA_DV_REALM_AUTH View	lists the authorization of a named database user account or database role (GRANTEE) to access realm objects in a particular realm
DBA_DV_REALM_OBJECT View	Lists the database schemas, or subsets of schemas with specific database objects contained therein, that are secured by the realms

6

Configuring Rule Sets

Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.

- What Are Rule Sets?
  A rule set is a collection of one or more rules.
- Rule Sets and Rules in a Multitenant Environment
  In a multitenant environment, you can create a rule set and its associated rules in the application root.
- Default Rule Sets
  Oracle Database Vault provides a set of default rules sets that you can customize for your needs.
- Creating a Rule Set
  To create a rule set, you first create the rule set itself, and then you can edit the rule set to associate it with one or more rules.
- Creating a Rule to Add to a Rule Set
  A rule defines the behavior that you want to control; a rule set is a named collection of rules.
- Removing Rule Set References to Oracle Database Vault Components
  Before you remove a rule set, you should remove the rule set references to Oracle Database Vault components.
- Deleting a Rule Set
  You can use Enterprise Manager Cloud Control to find reference to the rule set and then delete the rule set.
- How Rule Sets Work
  Understanding how rule sets work helps to create more effective rule sets.
- Tutorial: Creating an Email Alert for Security Violations
  This tutorial demonstrates how to use the `UTL_MAIL` PL/SQL package and an access control list to create an email alert for security violations.
- Tutorial: Configuring Two-Person Integrity, or Dual Key Security
  This tutorial demonstrates how to use Oracle Database Vault to control the authorization of two users.
- Guidelines for Designing Rule Sets
  Oracle provides guidelines for designing rule sets.
- How Rule Sets Affect Performance
  The number and complexity of rules can slow database performance.
- Rule Set and Rule Related Reports and Data Dictionary Views
  Oracle Database Vault provides reports and data dictionary views that are useful for analyzing rule sets and the rules within them.

What Are Rule Sets?

A rule set is a collection of one or more rules.

You can associate the rule set with a realm authorization, factor assignment, command rule, or secure application role.

The rule set evaluates to true or false based on the evaluation of each rule it contains and the evaluation type (*All True* or *Any True*). A rule within a rule set is a PL/SQL expression that evaluates to true or false. You can create a rule and add the rule to multiple rule sets.

You can use rule sets to accomplish the following activities:

- As a further restriction to realm authorization, to define the conditions under which realm authorization is active
- To define when to allow a command rule
- To enable a secure application role
- To define when to assign the identity of a factor

When you create a rule set, Oracle Database Vault makes it available for selection when you configure the authorization for a realm, command rule, factor, or secure application role.

Related Topics

- Rule Set and Rule Related Reports and Data Dictionary Views
  Oracle Database Vault provides reports and data dictionary views that are useful for analyzing rule sets and the rules within them.
- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.

Rule Sets and Rules in a Multitenant Environment

In a multitenant environment, you can create a rule set and its associated rules in the application root.

A common realm must use a common rule set when the associated realm or command rule is evaluated by Database Vault. The common rule set and its rules can only be created in the application root. After the common rule set is created, it exists in every container that is associated with the root where the common rule set is created. The common rule set can only include common rules.

To configure a common rule set and its rules, you must be commonly granted the `DV_OWNER` or `DV_ADMIN` role.

Related Topics

- Command Rules in a Multitenant Environment
  In a multitenant environment, you can create common and local command rules in either the CDB root or the application root.

Default Rule Sets

Oracle Database Vault provides a set of default rules sets that you can customize for your needs.

You can find a full list of rule sets by querying the `DBA_DV_RULE_SET` data dictionary view. To find rules that are associated with a rule set, query the `DBA_DV_RULE_SET_RULE` data dictionary view.

The default rule sets are as follows:

- **Allow Dumping Datafile Header:** This rule set prevents the dumping of data blocks.
- **Allow Fine Grained Control for Alter System:** This rule set enables you to control the ability of users to set initialization parameters using the `ALTER SYSTEM` SQL statement.
- **Allow Fine Grained Control of System Parameters: Note:** This rule set has been deprecated.

  This rule set provides a very flexible, fine-grained control over initialization parameters that manage system security, dump or destination location, backup and restore settings, optimizer settings, PL/SQL debugging, and security parameters. It affects the following initialization parameters, based on the associated rules of this rule set:

  – **Are System Security Parameters Allowed rule:** Cannot set `O7_DICTIONARY_ACCESSIBILITY`

  – **Are Dump or Dest Parameters Allowed rule:** Cannot set the following parameters:

Parameter B-D	Parameters D-U
`BACKGROUND_DUMP_DEST`	`DB_RECOVERY_FILE_DEST`
`CORE_DUMP_DEST`	`DIAGNOSTIC_DEST`
`DUMP_DATAFILE`	`LOG_ARCHIVE_MIN_SUCCEED_DEST`
`USER_DUMP_DEST`	`LOG_ARCHIVE_TRACE`
`DB_CREATE_ONLINE_LOG_DEST`	`USER_DUMP_DEST`

  – **Are Backup Restore Parameters Allowed rule:** Cannot set `RECYCLEBIN` (but does not prevent disabling the recycle bin)

  – **Are Database File Parameters Allowed rule:** Cannot set `CONTROL_FILES`

  – **Are Optimizer Parameters Allowed rule:** Can set `OPTIMIZER_SECURE_VIEW_MERGING = FALSE` (but `TRUE` not allowed)

  – **Are PL-SQL Parameters Allowed rule:** Can set `PLSQL_DEBUG = FALSE` (but `TRUE` not allowed).

  – **Are Security Parameters Allowed rule:** Cannot set the following:

Parameters A-A	Parameters O-S
`AUDIT_SYS_OPERATIONS = FALSE`	`OS_ROLES = TRUE`

Parameters A-A	Parameters O-S
AUDIT_TRAIL = NONE or FALSE	REMOTE_OS_ROLES = TRUE
AUDIT_SYSLOG_LEVEL	SQL92_SECURITY = FALSE

See *Oracle Database Reference* for detailed information about initialization parameters.

- **Allow Sessions:** Controls the ability to create a session in the database. This rule set enables you to add rules to control database logins using the CONNECT command rule. The CONNECT command rule is useful to control or limit SYSDBA access to programs that require its use. This rule set is not populated.
- **Allow System Parameters: Note:** This rule set has been deprecated.

  Controls the ability to set system initialization parameters. Since Oracle Database 11*g* Release 2 (11.2), the Allow Fine Grained Control of System Parameters rule set has replaced this rule set, but it is still supported for backward compatibility. The Allow System Parameters rule set is not associated with any commands, but its rules are still available and can be used with any custom rule set. Oracle recommends that you use the Allow Fine Grained Control of System Parameters rule set.
- **Can Grant VPD Administration:** Controls the ability to grant the GRANT EXECUTE or REVOKE EXECUTE privileges on the Oracle Virtual Private Database DBMS_RLS package, with the GRANT and REVOKE statements.
- **Can Maintain Accounts/Profiles:** Controls the roles that manage user accounts and profiles, through the CREATE USER, DROP USER, CREATE PROFILE, ALTER PROFILE, or DROP PROFILE statements.
- **Can Maintain Own Account:** Allows the accounts with the DV_ACCTMGR role to manage user accounts and profiles with the ALTER USER statement. Also allows individual accounts to change their own password using the ALTER USER statement. See DV_ACCTMGR Database Vault Account Manager Role for more information about the DV_ACCTMGR role.
- **Disabled:** Convenience rule set to quickly disable security configurations like realms, command rules, factors, and secure application roles.
- **Enabled:** Convenience rule set to quickly enable system features.
- **Not allow to set AUDIT_SYS_OPERATIONS to False:** Prevents the AUDIT_SYS_OPERATIONS initialization parameter from being set to FALSE. If unified auditing is enabled, then the AUDIT_SYS_OPERATIONS parameter has no effect.
- **Not allow to set OPTIMIZER_SECURE_VIEW_MERGING to True:** Prevents the OPTIMIZER_SECURE_VIEW_MERGING initialization parameter from being set to TRUE.
- **Not allow to set OS_ROLES to True:** Prevents the OS_ROLES initialization parameter from being set to TRUE.
- **Not allow to set PLSQL_DEBUG to True:** Prevents the PLSQL_DEBUG initialization parameter from being set to TRUE.
- **Not allow to set REMOTE_OS_ROLES to True:** Prevents the REMOTE_OS_ROLES initialization parameter from being set to TRUE.
- **Not allow to set SQL92_SECURITY to False:** Prevents the SQL92_SECURITY from being set to FALSE.
- **Not allow to turn off AUDIT_TRAIL:**

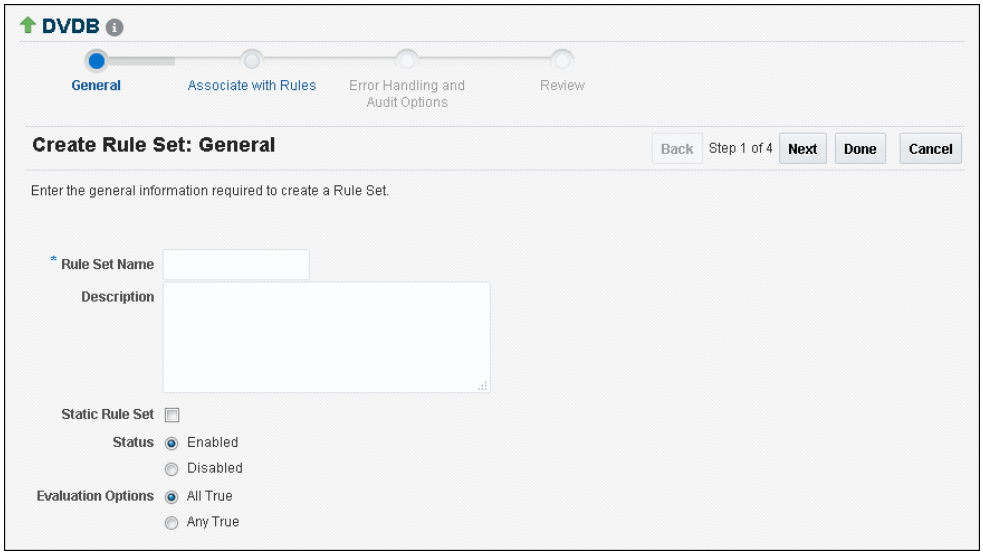
Prevents the AUDIT_TRAIL initialization parameter from being turned off. If unified auditing is enabled, then the AUDIT_TRAIL parameter has no effect.

Creating a Rule Set

To create a rule set, you first create the rule set itself, and then you can edit the rule set to associate it with one or more rules.

You can associate a new rule with the rule set, add existing rules to the rule set, or delete a rule association from the rule set.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the DV_OWNER or DV_ADMIN role and the SELECT ANY DICTIONARY privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Rule Sets**.
3. In the Rule Sets page, click **Create** to display the Create Rule Sets page.



4. In the General page, enter the following information:
   - **Name:** Enter a name for the rule set. It can contain up to 90 characters in mixed-case. Spaces are allowed. This attribute is mandatory.

     Oracle suggests that you start the name with a verb and complete it with the realm or command rule name to which the rule set is attached. For example:

     ```
     Limit SQL*Plus access
     ```

   - **Description:** Enter a description of the functionality for the rule set. It can have up to 1024 characters in mixed-case. This attribute is optional.

     You may want to document the business requirement of the rule set. For example:

     ```
     Rule to limit access to SQL*Plus
     ```

- **Static Rule Set:** You can control how often the rule set is evaluated when it is accessed during a user session. A static rule set is evaluated once when accessed for the first time in a user session. After that, the evaluated value is re-used in the user session. On the other hand, a non-static rule set is evaluated every time it is accessed.
- **Status:** Select either **Enabled** or **Disabled** to enable or disable the rule set during run time. This attribute is mandatory.
- **Evaluation Options:** If you plan to assign multiple rules to a rule set, then select one of the following settings:
  - **All True:** All rules in the rule set must evaluate to true for the rule set itself to evaluate to true.
  - **Any True:** At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to true.

5. Click **Next** to display the Associate with Rules page.
6. Select one of the following options:
   - **Add Existing Rule:** Double-click from the list of available rules to move them to the Selected Rules list, and then click **OK**.
   - **Create Rule:** Enter a name and `WHERE` clause expression that evaluates to true or false. Click **OK**. See Creating a Rule to Add to a Rule Set for more information.
7. Click **Next** to display the Error handling and Audit options page.
8. Enter the following information:
   - **Error Handling:** Select either **Show Error Message** or **Do Not Show Error Message**.

     An advantage of selecting **Do Not Show Error Message** and then enabling auditing is that you can track the activities of a potential intruder. The audit report reveals the activities of the intruder, yet the intruder is unaware that you are doing this because he or she does not see any error messages.
   - **Fail Code:** Enter a number in the ranges of -20000 to -20999 or 20000 to 20999. The error code is displayed with the **Fail Message** (created next) when the rule set evaluates to false or one of the associated rules contains an invalid PL/SQL expression. If you omit this setting, then Oracle Database Vault displays a generic error code.
   - **Fail Message:** Enter a message, up to 80 characters in mixed-case, to associate with the fail code you specified under **Fail Code**. The error message is displayed when the rule set evaluates to false or one of the associated rules contains an invalid PL/SQL expression. If you do not specify an error message, then Oracle Database Vault displays a generic error message.
   - **Custom Event Handler Option:** Select one of the following options to determine when to run the **Custom Event Handler Logic** (created next).
     - **Handler Disabled:** Does not run any custom event method.
     - **Execute On Failure:** Runs the custom event method when the rule set evaluates to false or one of the associated rules contains an invalid PL/SQL expression.
     - **Execute On Success:** Runs the custom event method when the rule set evaluates to true.

You can create a custom event method to provide special processing outside the standard Oracle Database Vault rule set auditing features. For example, you can use an event handler to initiate a workflow process or send event information to an external system.

- **Custom Event Handler Logic:** Enter a PL/SQL expression up to 255 characters in mixed-case. An expression may include any package procedure or standalone procedure. You can create your own expression or use the PL/SQL interfaces described in Oracle Database Vault Rule Set APIs.

  Write the expression as a fully qualified procedure (such as *schema*.*procedure_name*). Do not include any other form of SQL statements. If you are using application package procedures or standalone procedures, you must provide DVSYS with the EXECUTE privilege on the object. The procedure signature can be in one of the following two forms:

  – PROCEDURE *my_ruleset_handler*(*p_ruleset_name* IN VARCHAR2, *p_ruleset_rules* IN BOOLEAN): Use this form when the name of the rule set and its return value are required in the handler processing.

  – PROCEDURE *my_ruleset_handler*: Use this form when the name of the rule set and its return value are not required in the handler processing.

  Be aware that you cannot use invoker's rights procedures as event handlers. Doing so can cause the rule set evaluation to fail unexpectedly. Only use definer's rights procedures as event handlers.

  Use the following syntax:

  myschema.*my_ruleset_handler*

- **Audit Options:** Select from the following options to generate an audit record for the rule set in a non-unified auditing environment. Oracle Database Vault writes the audit trail to the DVSYS.AUDIT_TRAIL$ table. (If you have enabled unified auditing, then this setting does not capture audit records. Instead, you must create unified audit policies to capture this information.)

  – **Audit Disabled:** Does not create an audit record under any circumstances.

  – **Audit on Success:** Creates an audit record when the rule set evaluates to true.

  – **Audit On Failure:** Creates an audit record when the rule set evaluates to false or one of the associated rules contains an invalid PL/SQL expression.

  – **Audit On Success or Failure:** Creates an audit record whenever a rule set is evaluated.

9. Click **Next** to display the Review page.
10. Review the settings, and if they are satisfactory, click **Finish**.

See Also:

- Auditing Oracle Database Vault for more information about audit records in the `DVSYS.AUDIT_TRAIL$` table
- *Oracle Database Security Guide* for information about creating unified audit policies for Database Vault

Creating a Rule to Add to a Rule Set

A rule defines the behavior that you want to control; a rule set is a named collection of rules.

- About Creating Rules
  You can create rules during the rule set creation process, or independently of it.
- Default Rules
  Default rules are rules that have commonly used behavior, such as checking if an action evaluates to true or false.
- Creating a New Rule
  You can create a new rule in Enterprise Manager Cloud Control.
- Adding Existing Rules to a Rule Set
  After you have created one or more rules, you can use Enterprise Manager Cloud Control to add to a rule set.
- Removing a Rule from a Rule Set
  Before you remove a rule from a rule set, you can locate the various references to it using Cloud Control.

About Creating Rules

You can create rules during the rule set creation process, or independently of it.

After you create the rule, you can associate a rule set with one or more additional rules.

If you create a new rule during the rule set creation process, the rule is automatically added to the current rule set. You also can add existing rules to the rule set. Alternatively, you can omit adding rules to the rule set and use it as a template for rule sets you may want to create in the future.

You can add as many rules that you want to a rule set, but for better design and performance, you should keep the rule sets simple. See Guidelines for Designing Rule Sets for additional advice.

The rule set evaluation depends on the evaluation of its rules using the Evaluation Options (**All True** or **Any True**). If a rule set is disabled, Oracle Database Vault evaluates the rule set to true without evaluating its rules.

Related Topics

- How Rule Sets Work
  Understanding how rule sets work helps to create more effective rule sets.

Default Rules

Default rules are rules that have commonly used behavior, such as checking if an action evaluates to true or false.

You can find a full list of rules by querying the DBA_DV_RULE data dictionary view.Table 6-1 lists the current default Oracle Database rules.

Table 6-1 Current Default Oracle Database Vault Rules

Rule	Description
Are Backup Restore Parameters Allowed	Note: This default rule has been deprecated. Checks if the current SQL statement attempts to turn on the RECYCLEBIN parameter
Are Database File Parameters Allowed	Note: This default rule has been deprecated. Checks if the current SQL statement attempts to alter control file related configuration
Are Dump Parameters Allowed	Checks if the current SQL statement attempts to alter initialization parameters related to the destination of a dump
Are Dest Parameters Allowed	Checks if the current SQL statement attempts to alter initialization parameters related to the size limit of a dump
Are Dump or Dest Parameters Allowed	Note: This default rule has been deprecated. Checks if the current SQL statement attempts to alter initialization parameters related to the size limit or destination of dump
Are Optimizer Parameters Allowed	Note: This default rule has been deprecated. Checks if the current SQL statement attempts to alter the setting for the OPTIMIZER_SECURE_VIEW_MERGING parameter
Are PL-SQL Parameters Allowed	Note: This default rule has been deprecated. Checks if the current SQL statement attempts to alter the PLSQL_DEBUG initialization parameter.
Are Security Parameters Allowed	Note: This default rule has been deprecated. Checks if there is an attempt to disable the following initialization parameters: • AUDIT_SYS_OPERATIONS • AUDIT_TRAIL • AUDIT_SYSLOG_LEVEL • SQL92_SECURITY Note that if you have enabled unified auditing, then the AUDIT_SYS_OPERATIONS, AUDIT_TRAIL, and AUDIT_SYSLOG_LEVEL parameters have no effect. This rule prevents any attempt to enable the following parameters: • OS_ROLES • REMOTE_OS_ROLES

Table 6-1 (Cont.) Current Default Oracle Database Vault Rules

Rule	Description
Are System Security Parameters Allowed	Note: This default rule has been deprecated. Prevents modification of the following parameters: • O7_DICTIONARY_ACCESSIBILITY • DYNAMIC_RLS_POLICIES • _SYSTEM_TRIG_ENABLED
False	Evaluates to FALSE
Is Alter DVSYS Allowed	Note: This default rule has been deprecated. Checks if the logged-in user can execute the ALTER USER statement on other users successfully
Is Database Administrator	Checks if a user has been granted the DBA role
Is Drop User Allowed	Checks if the logged in user can drop users
Is Dump of Block Allowed	Checks if the dumping of blocks is allowed
Is First Day of Month	Checks if the specified date is the first day of the month
Is Label Administrator	Checks if the user has been granted the LBAC_DBA role
Is Last Day of Month	Checks if the specified date is the last day of the month
Is _dynamic_rls_init Parameters Allowed	Note: This default rule has been deprecated. Prevent modification of the DYNAMIC_RLS_POLICIES parameter
Is Parameter Value False	Checks if a specified parameter value has been set to FALSE
Is Parameter Value None	Checks if a specified parameter value has been set to NONE
Is Parameter Value Not False	Checks if a specified parameter value has been set to <> FALSE
Is Parameter Value Not None	Checks if a specified parameter value has been set to <> NONE
Is Parameter Value Not Off	Checks if a specified parameter value has been set to <> OFF
Is Parameter Value Not On	Checks if a specified parameter value has been set to <> ON
Is Parameter Value Not True	Checks if a specified parameter value has been set to <> TRUE
Is Parameter Value Off	Checks if a specified parameter value has been set to OFF
Is Parameter Value On	Checks if a specified parameter value has been set to ON
Is Parameter Value True	Checks if a specified parameter value has been set to TRUE
Is SYS or SYSTEM User	Checks if the user is SYS or SYSTEM
Is Security Administrator	Checks if a user has been granted the DV_ADMIN role
Is Security Owner	Checks if a user has been granted the DV_OWNER role

Table 6-1 (Cont.) Current Default Oracle Database Vault Rules

Rule	Description
Is User Manager	Checks if a user has been granted the DV_ACCTMGR role
Is _system_trig_enabled Parameters Allowed	Note: This default rule has been deprecated. Checks if the user tries to modify the following system parameters, but in database recovery operations, this rule permits these parameters to be changed. • AUDIT_SYS_OPERATIONS: Prevents users from setting it to FALSE • AUDIT_TRAIL: Prevents users from setting it to NONE or FALSE • AUDIT_SYSLOG_LEVEL: Blocks all operations on this parameter • CONTROL_FILES: Blocks all operations • OPTIMIZER_SECURE_VIEW_MERGING: Prevents users from setting it to TRUE • OS_ROLES: Prevents users from setting it to TRUE • PLSQL_DEBUG: Prevents users from setting it to ON • RECYCLEBIN: Prevents users from setting it to ON • REMOTE_OS_ROLES: Prevents users from setting it to TRUE • SQL92_SECURITY: Prevents users from setting it to FALSE Note that if you have enabled unified auditing, then the AUDIT_SYS_OPERATIONS, AUDIT_TRAIL, and AUDIT_SYSLOG_LEVEL parameters have no effect.
Is o7_dictionary_accessibility Parameters Allowed	Note: This default rule has been deprecated. Checks if current SQL statement attempts to alter the setting of the O7_DICTIONARY_ACCESSIBILITY parameter
Login User Is Object User	Checks if the logged in user is the same as the user about to be altered by the current SQL statement
No Exempt Access Policy Role	Checks if the user has been granted the EXEMPT ACCESS POLICY role or is user SYS
Not Export Session	Obsolete
True	Evaluates to TRUE

Creating a New Rule

You can create a new rule in Enterprise Manager Cloud Control.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the DV_OWNER or DV_ADMIN role and the SELECT ANY DICTIONARY privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Rules**.

3. Click the **Create** button.
4. In the Create Rule page, enter the following settings:
   - **Name:** Enter a name for the rule. Use up to 90 characters in mixed-case.

     Oracle suggests that you start the name with a verb and complete the name with the purpose of the rule. For example:

     ```
     Prevent non-admin access to SQL*Plus
     ```

     Because rules do not have a **Description** field, make the name explicit but be sure to not exceed over 90 characters.
   - **Rule Expression:** Enter a PL/SQL expression that fits the following requirements:
     - It is valid in a SQL `WHERE` clause.
     - It can be a freestanding and valid PL/SQL Boolean expression such as the following:

       ```
       TO_CHAR(SYSDATE,'HH24') = '12'
       ```
     - It must evaluate to a Boolean (`TRUE` or `FALSE`) value.
     - It must be no more than 1024 characters long.
     - It can contain existing and compiled PL/SQL functions from the current database instance. Ensure that these are fully qualified functions (such as *`schema`*.*`function_name`*). Do not include any other form of SQL statements.

       Be aware that you cannot use invoker's rights procedures with rule expressions. Doing so will cause the rule evaluation to fail unexpectedly. Only use definer's rights procedures with rule expressions.

       If you want to use application package functions or standalone functions, you must grant the `DVSYS` account the `EXECUTE` privilege on the function. Doing so reduces the chances of errors when you add new rules.
     - Ensure that the rule works. You can test the syntax by running the following statement in SQL*Plus:

       ```
       SELECT rule_expression FROM DUAL;
       ```

       For example, suppose you have created the following the rule expression:

       ```
       SYS_CONTEXT('USERENV','SESSION_USER') != 'TSMITH'
       ```

       You could test this expression as follows:

       ```
       SELECT SYS_CONTEXT('USERENV','SESSION_USER') FROM DUAL;
       ```

       For the Boolean example listed earlier, you would enter the following:

       ```
       SELECT TO_CHAR(SYSDATE,'HH24')FROM DUAL;
       ```
5. Click **OK**.

Related Topics

- Oracle Database Vault PL/SQL Rule Set Functions
  Oracle Database Vault provides functions to use in rule sets to inspect the SQL statement that the rule set protects.

- DBMS_MACADM Rule Set Procedures
  The DBMS_MACADM rule set procedures enable you to configure both rule sets and individual rules that go within these rule sets.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the DBMS_MACUTL PL/SQL package.

Adding Existing Rules to a Rule Set

After you have created one or more rules, you can use Enterprise Manager Cloud Control to add to a rule set.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the DV_OWNER or DV_ADMIN role and the SELECT ANY DICTIONARY privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Rule Sets**.
3. Select the rule set to which you want to add an existing rule, and then click **Edit**.
4. Click **Next** until you reach the Associate with Rules page.
5. Click **Add Existing Rule** to display the Add Existing Rules dialog box.
6. In the Add Existing Rules page, select the rules you want, and then click **Move** (or **Move All**, if you want all of them) to move them to the Selected Rules list.

   You can select multiple rules by holding down the **Ctrl** key as you click each rule.
7. Click **OK**.
8. Click **Done**, then click **Finish**.

Removing a Rule from a Rule Set

Before you remove a rule from a rule set, you can locate the various references to it using Cloud Control.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the DV_OWNER or DV_ADMIN role and the SELECT ANY DICTIONARY privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Rule Sets**.

   If you are not sure which rule set contains that rule that you want to remove, then select **Rules** from Database Vault Components, select the rule that you want to remove, and then select the **View** option (but not the **View** menu). The rule sets that are associated with the rule are listed in Rule Set Usages.
3. Select the rule set to which you want to add an existing rule, and then click **Edit**.
4. Click **Next** until you reach the Associate with Rules page.
5. Select the rule you want to delete and click **Remove**.
6. Click **Done**, then click **Finish**.

After you remove the rule from the rule set, the rule still exists. If you want, you can associate it with other rule sets. If you want to delete the rule, then you can do so from the Rules page.

Removing Rule Set References to Oracle Database Vault Components

Before you remove a rule set, you should remove the rule set references to Oracle Database Vault components.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. Find the references to the rule set that you want to delete.

   In the Rule Sets page, select the rule set and then click the **View** button (but not the **View** menu). In the View Rule Set page, check the Ruleset Usages area for the references to the rule set that you want to remove. Click **OK**.
3. In the Administration page, under Database Vault Components, select the component that contains the reference to the rule set (such as **Realms**).
4. Select the object, and then click **Edit**.
5. Click **Next** until you reach the authorizations page.
6. Select the authorization with the rule set and then click **Edit**, and then remove the referenced object.
7. Click **Done**, then click **Finish**.

Deleting a Rule Set

You can use Enterprise Manager Cloud Control to find reference to the rule set and then delete the rule set.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. Remove references to the rule set.
3. Select the rule set that you want to remove and click **Delete**.
4. In the Confirmation window, click **Yes**.

   The rule set is deleted. Optionally, you can choose to remove the existing associations with rules before deleting the rule set.

Related Topics

- Removing Rule Set References to Oracle Database Vault Components
  Before you remove a rule set, you should remove the rule set references to Oracle Database Vault components.

How Rule Sets Work

Understanding how rule sets work helps to create more effective rule sets.

- How Oracle Database Vault Evaluates Rules
  Oracle Database Vault evaluates the rules within a rule set as a collection of expressions.
- Nested Rules within a Rule Set
  You can nest one or more rules within the rule set.
- Creating Rules to Apply to Everyone Except One User
  You can also create rules to apply to everyone *except* one user (for example, a privileged user).

How Oracle Database Vault Evaluates Rules

Oracle Database Vault evaluates the rules within a rule set as a collection of expressions.

If you have set **Evaluation Options** to **All True** and if a rule evaluates to false, then the evaluation stops at that point, instead of attempting to evaluate the rest of the rules in the rule set. Similarly, if **Evaluation Options** is set to **Any True** and if a rule evaluates to true, the evaluation stops at that point. If a rule set is disabled, Oracle Database Vault evaluates it to true without evaluating its rules.

Nested Rules within a Rule Set

You can nest one or more rules within the rule set.

For example, suppose you want to create a nested rule, Is Corporate Network During Maintenance, that performs the following two tasks:

- It limits table modifications only when the database session originates within the corporate network.
- It restricts table modifications during the system maintenance window scheduled between 10:00 p.m. and 10:59 p.m.

The rule definition would be as follows:

```
DVF.F$NETWORK = 'Corporate' AND TO_CHAR(SYSDATE,'HH24') between '22' AND '23'
```

Related Topics

- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the `DVF` schema functions when you use the `DBMS_MACADM` PL/SQL package to manage the various factors.
- Configuring Factors
  Factors enable you to base Database Vault restrictions on attributes such as a client IP address or a domain.

Creating Rules to Apply to Everyone Except One User

You can also create rules to apply to everyone *except* one user (for example, a privileged user).

- To create a rule that excludes specific users, user the `SYS_CONTEXT` function.

For example:

```
SYS_CONTEXT('USERENV','SESSION_USER') = 'SUPERADMIN_USER' OR additional_rule
```

If the current user is a privileged user, then the system evaluates the rule to true without evaluating *additional_rule*. If the current user is not a privileged user, then the evaluation of the rule depends on the evaluation of *additional_rule*.

Tutorial: Creating an Email Alert for Security Violations

This tutorial demonstrates how to use the UTL_MAIL PL/SQL package and an access control list to create an email alert for security violations.

- About This Tutorial
  In tutorial, you create an email alert that is sent when a user attempts to alter a table outside a maintenance period.
- Step 1: Install and Configure the UTL_MAIL PL/SQL Package
  The UTL_MAIL PL/SQL package, which you must manually install, has procedures to manage email notifications.
- Step 2: Create an Email Security Alert PL/SQL Procedure
  User leo_dvowner can use the CREATE PROCEDURE statement to create the email security alert.
- Step 3: Configure an Access Control List File for Network Services
  Before you can use UTL_MAIL, you must configure an access control list (ACL) to enable fine-grained access to external network services.
- Step 4: Create a Rule Set and a Command Rule to Use the Email Security Alert
  To create the rule set and command rule, you can use DBMS_MACADM PL/SQL package.
- Step 5: Test the Email Security Alert
  After the alert has been created, it is ready to be tested.
- Step 6: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

About This Tutorial

In tutorial, you create an email alert that is sent when a user attempts to alter a table outside a maintenance period.

To do this, you must create a rule to set the maintenance period hours, attach this rule to a rule set, and then create a command rule to allow the user to alter the table. You then associate the rule set with this command rule, which then sends the email alert when the user attempts to use the ALTER TABLE SQL statement outside the maintenance period.

> **Note:**
>
> To complete this tutorial, you must use a database that has an SMTP server.

Step 1: Install and Configure the UTL_MAIL PL/SQL Package

The UTL_MAIL PL/SQL package, which you must manually install, has procedures to manage email notifications.

1. Log into the database instance as SYS using the SYSDBA administrative privilege.

   ```
   sqlplus sys as sysdba
   Enter password: password
   ```

2. In a multitenant environment, connect to the appropriate pluggable database (PDB).

   For example:

   ```
   CONNECT SYS@my_pdb AS SYSDBA
   Enter password: password
   ```

   To find the available PDBs, run the `show pdbs` command. To check the current PDB, run the `show con_name` command.

3. Install the UTL_MAIL package.

   ```
   @$ORACLE_HOME/rdbms/admin/utlmail.sql
   @$ORACLE_HOME/rdbms/admin/prvtmail.plb
   ```

   The UTL_MAIL package enables you to manage email. See *Oracle Database PL/SQL Packages and Types Reference* for more information about UTL_MAIL. However, be aware that currently, the UTL_MAIL PL/SQL package do not support SSL servers.

4. Check the current value of the SMTP_OUT_SERVER parameter, and make a note of this value so that you can restore it when you complete this tutorial.

   For example:

   ```
   SHOW PARAMETER SMTP_OUT_SERVER
   ```

   Output similar to the following appears:

   ```
   NAME                 TYPE             VALUE
   -------------------- ---------------- --------------------------------
   SMTP_OUT_SERVER      string           some_value.example.com
   ```

5. Issue the following ALTER SYSTEM statement:

   ```
   ALTER SYSTEM SET SMTP_OUT_SERVER="imap_mail_server.example.com";
   ```

   Replace *imap_mail_server.example.com* with the name of your SMTP server, which you can find in the account settings in your email tool. Enclose these settings in double quotation marks. For example:

   ```
   ALTER SYSTEM SET SMTP_OUT_SERVER="my_imap_mail_server.example.com"
   ```

6. Connect as SYS using the SYSOPER privilege and then restart the database.

   ```
   CONNECT SYS AS SYSOPER -- Or, CONNECT SYS@hrpdb AS SYSOPER
   Enter password: password

   SHUTDOWN IMMEDIATE
   STARTUP
   ```

7. Ensure that the SMTP_OUT_SERVER parameter setting is correct.

```
CONNECT SYS AS SYSDBA -- Or, CONNECT SYS@hrpdb AS SYSDBA
Enter password: password

SHOW PARAMETER SMTP_OUT_SERVER
```

Output similar to the following appears:

```
NAME                 TYPE             VALUE
-------------------- ---------------- --------------------------------
SMTP_OUT_SERVER      string           my_imap_mail_server.example.com
```

Step 2: Create an Email Security Alert PL/SQL Procedure

User `leo_dvowner` can use the `CREATE PROCEDURE` statement to create the email security alert.

1. Ensure that you are connected as a user who has privileges to perform the grants described in this step, and then grant these privileges to a user who has been granted the `DV_OWNER` role. You should also be authorized as an owner of the Oracle System Privilege and Role Management realm.

   (Alternatively, you can select a user who has been granted the `DV_ADMIN` role, but for this tutorial, you will select a user who has the `DV_OWNER` role.)

   For example:

   ```
   CONNECT dba_psmith -- Or, CONNECT dba_psmith@hrpdb
   Enter password: password

   GRANT CREATE PROCEDURE, DROP ANY PROCEDURE TO leo_dvowner;
   GRANT EXECUTE ON UTL_TCP TO leo_dvowner;
   GRANT EXECUTE ON UTL_SMTP TO leo_dvowner;
   GRANT EXECUTE ON UTL_MAIL TO leo_dvowner;
   GRANT EXECUTE ON DBMS_NETWORK_ACL_ADMIN TO leo_dvowner;
   ```

   The `UTL_TCP`, `UTL_SMTP`, `UTL_MAIL`, and `DBMS_NETWORK_ACL_ADMIN` PL/SQL packages will be used by the email security alert that you create.

2. Connect to SQL*Plus as the `DV_OWNER` user.

   For example:

   ```
   CONNECT leo_dvowner -- Or, CONNECT leo_dvowner@hrpdb
   Enter password: password
   ```

3. Create the following procedure:

   ```
   CREATE OR REPLACE PROCEDURE email_alert AS
   msg varchar2(20000) := 'Realm violation occurred for the ALTER TABLE Command
   Security Policy rule set. The time is: ';
   BEGIN
     msg := msg||to_char(SYSDATE, 'Day DD MON, YYYY HH24:MI:SS');
   UTL_MAIL.SEND (
       sender      => 'youremail@example.com',
       recipients  => 'recipientemail@example.com',
       subject     => 'Table modification attempted outside maintenance!',
       message     => msg);
   END email_alert;
   /
   ```

Replace *youremail@example.com* with your email address, and *recipientemail@example.com* with the email address of the person you want to receive the notification.

4. Grant the EXECUTE permission on this procedure to DVSYS.

```
GRANT EXECUTE ON email_alert TO DVSYS;
```

Step 3: Configure an Access Control List File for Network Services

Before you can use UTL_MAIL, you must configure an access control list (ACL) to enable fine-grained access to external network services.

For detailed information about fine-grained access to external network services, see *Oracle Database Security Guide*.

1. As the DV_OWNER user, in SQL*Plus, configure the following access control setting and its privilege definitions.

```
BEGIN
 DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
  host       => 'SMTP_OUT_SERVER_setting',
  lower_port => 25,
  ace        =>  xs$ace_type(privilege_list => xs$name_list('smtp'),
                             principal_name => 'LEO_DVOWNER,
                             principal_type => xs_acl.ptype_db));
END;
/
```

In this example:

- *lower_port*: Enter the port number that your email tool specifies for its outgoing server. Typically, this setting is 25. Enter this value for both the lower_port and upper_port settings. (Currently, the UTL_MAIL package does not support SSL. If your mail server is an SSL server, then enter 25 for the port number, even if the mail server uses a different port number.)
- principal_name: Replace LEO_DVOWNER with the name of the DV_OWNER user.
- host: For the *SMTP_OUT_SERVER_setting*, enter the SMTP_OUT_SERVER setting that you set for the SMTP_OUT_SERVER parameter in Step 1: Install and Configure the UTL_MAIL PL/SQL Package. This setting should match exactly the setting that your email tool specifies for its outgoing server.

2. Commit your changes to the database.

```
COMMIT;
```

3. Test the settings that you have created so far.

```
EXEC EMAIL_ALERT;
COMMIT;
```

SQL*Plus should display a PL/SQL procedure successfully completed message, and in a moment, depending on the speed of your email server, you should receive the email alert.

If you receive an ORA-24247: network access denied by access control list (ACL) error followed by ORA-06512: at *string*line *string* errors, then check the settings in the access control list file.

Step 4: Create a Rule Set and a Command Rule to Use the Email Security Alert

To create the rule set and command rule, you can use DBMS_MACADM PL/SQL package.

1. As the DV_OWNER user, create the following rule set:

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'ALTER TABLE Command Security Policy',
  description      => 'This rule set allows ALTER TABLE only during the
                       maintenance period.',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
  fail_message     => '',
  fail_code        => NULL,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler          => 'leo_dvowner.email_alert');
END;
/
```

2. Create a rule similar to the following.

   For now, set the rule expression to be during the time you test it. For example, if you want to test it between 2 p.m. and 3 p.m., create the rule as follows:

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Restrict Access to Maintenance Period',
  rule_expr  => 'TO_CHAR(SYSDATE,''HH24'') BETWEEN ''14'' AND ''15''');
END;
/
```

   Ensure that you use two single quotation marks instead of double quotation marks for HH24, 14, and 15.

   You can check the system time on your computer by issuing the following SQL statement:

```
SELECT TO_CHAR(SYSDATE,'HH24') FROM DUAL;
```

   Output similar to the following appears:

```
TO
--
14
```

   Later on, when you are satisfied that the rule works, you can update it to a time when your site typically performs maintenance work (for example, between 7 p.m. and 10 p.m), as follows:

```
BEGIN
 DBMS_MACADM.UPDATE_RULE(
  rule_name  => 'Restrict Access to Maintenance Period',
  rule_expr  => 'TO_CHAR(SYSDATE,''HH24'') BETWEEN ''16'' AND ''22''');
END;
/
```

3. Add the Restrict Access to Maintenance Period rule to the ALTER TABLE Command Security Policy rule set.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'ALTER TABLE Command Security Policy',
  rule_name     => 'Restrict Access to Maintenance Period');
END;
/
```

4. Create the following command rule:

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'ALTER TABLE',
  rule_set_name   => 'ALTER TABLE Command Security Policy',
  object_owner    => 'SCOTT',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

5. Commit these updates to the database.

```
COMMIT;
```

Step 5: Test the Email Security Alert

After the alert has been created, it is ready to be tested.

1. Connect to SQL*Plus as user SCOTT.

   For example:

   ```
   CONNECT SCOTT -- Or, CONNECT SCOTT@hrpdb
   Enter password: password
   ```

   If the SCOTT account is locked and expired, then a user with the DV_ACCTMGR role can unlock this account and create a new password as follows:

   ```
   ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

2. As the user SCOTT, create a test table.

   ```
   CREATE TABLE mytest (col1 number);
   ```

3. Change the system time on your computer to a time when the ALTER TABLE Command Security Policy rule set takes place.

   For example, if you set the test period time to between 2 p.m. and 3 p.m., do the following:

   UNIX: Log in as root and use the date command to set the time. For example, assuming the date today is August 15, 2012, you would enter the following:

   ```
   $ su root
   Password: password

   $ date -s "08/15/2012 14:48:00"
   ```

Windows: Double-click the clock icon, which is typically at the lower right corner of the screen. In the Date and Time Properties window, set the time to 2 p.m., and then click **OK**.

4. Try altering the `my_test` table.

```
ALTER TABLE mytest ADD (col2 number);

Table altered.
```

`SCOTT` should be able to alter the `mytest` table during this time.

5. Reset the system time to a time outside the Restrict Access to Maintenance Period time.
6. Log in as `SCOTT` and try altering the `my_test` table again.

```
CONNECT SCOTT -- Or, CONNECT SCOTT@hrpdb
Enter password: password

ALTER TABLE mytest ADD (col3 number);
```

The following output should appear:

```
ORA-47400: Command Rule violation for ALTER TABLE on SCOTT.MYTEST
```

`SCOTT` cannot alter the `mytest` table. In a moment, you should receive an email with the subject header `Table modification attempted outside maintenance!` and with a message similar to the following:

```
Realm violation occurred for the ALTER TABLE Command Security Policy rule set.
The time is: Wednesday 15 AUG, 2012 14:24:25
```

7. Reset the system time to the correct time.

Step 6: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. Connect to SQL*Plus as the `DV_OWNER` user.

```
CONNECT leo_dvowner -- Or, CONNECT leo_dvowner@hrpdb
Enter password: password
```

2. In the order shown, drop the Oracle Database Vault rule components.

```
EXEC DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('ALTER TABLE Command Security
Policy', 'Restrict Access to Maintenance Period');
EXEC DBMS_MACADM.DELETE_RULE('Restrict Access to Maintenance Period');
EXEC DBMS_MACADM.DELETE_COMMAND_RULE('ALTER TABLE', 'SCOTT', '%');
EXEC DBMS_MACADM.DELETE_RULE_SET('ALTER TABLE Command Security Policy');
```

3. Drop the `email_alert` PL/SQL procedure.

```
DROP PROCEDURE email_alert;
```

4. Connect as user `SCOTT` and remove the `mytest` table.

```
CONNECT SCOTT -- Or, CONNECT SCOTT@hrpdb
Enter password: password

DROP TABLE mytest;
```

5. Connect as a user who has privileges to revoke privileges from other users.

   For example:

   ```
   CONNECT dba_psmith -- Or, CONNECT dba_psmith@hrpdb
   Enter password: password
   ```

6. Revoke the EXECUTE privilege on the UTL_TCP, UTL_SMTP, and UTL_MAIL PL/SQL packages from the DV_OWNER user.

   For example:

   ```
   REVOKE EXECUTE ON UTL_TCP FROM leo_dvowner;
   REVOKE EXECUTE ON UTL_SMTP FROM leo_dvowner;
   REVOKE EXECUTE ON UTL_MAIL FROM leo_dvowner;
   REVOKE EXECUTE ON DBMS_NETWORK_ACL_ADMIN FROM leo_dvowner;
   ```

7. Set the SMTP_OUT_SERVER parameter to its original value.

   For example:

   ```
   ALTER SYSTEM SET SMTP_OUT_SERVER="some_value.example.com";
   ```

8. Connect as SYS with the SYSOPER administrative privilege and then restart the database.

   ```
   CONNECT SYS AS SYSOPER -- Or, CONNECT SYS@hrpdb AS SYSOPER
   Enter password: password

   SHUTDOWN IMMEDIATE
   STARTUP
   ```

Tutorial: Configuring Two-Person Integrity, or Dual Key Security

This tutorial demonstrates how to use Oracle Database Vault to control the authorization of two users.

- About This Tutorial
  In this tutorial, you configure a rule set that defines two-person integrity (TPI).
- Step 1: Create Users for This Tutorial
  You must create two users for this tutorial, patch_boss and patch_user.
- Step 2: Create a Function to Check if User patch_boss Is Logged In
  The behavior of the Database Vault settings will be determined by the function.
- Step 3: Create Rules, a Rule Set, and a Command Rule to Control User Access
  Next, you must create two rules, a rule set to which you will add them, and a command rule.
- Step 4: Test the Users' Access
  After the rules have been created, they are ready to be tested.
- Step 5: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

About This Tutorial

In this tutorial, you configure a rule set that defines two-person integrity (TPI).

This feature is also called dual key security, dual key connection, and two-man rule security. In this type of security, two users are required to authorize an action instead of one user.

The idea is that one user provides a safety check for the other user before that user can proceed with a task. Two-person integrity provides an additional layer of security for actions that potentially can be dangerous. This type of scenario is often used for tasks such as database patch updates, which is what this tutorial will demonstrate. One user, `patch_user` must log into perform a database patch upgrade, but the only way that he can do this is if his manager, `patch_boss` is already logged in. You will create a function, rules, a rule set, and a command rule to control `patch_user`'s ability to log in.

Step 1: Create Users for This Tutorial

You must create two users for this tutorial, `patch_boss` and `patch_user`.

- `patch_boss` acts in a supervisory role: If `patch_boss` is not logged in, then the `patch_user` user cannot log in.
- `patch_user` is the user who is assigned to perform the patch upgrade. However, for this tutorial, user `patch_user` does not actually perform a patch upgrade. He only attempts to log in.

To create the users:

1. Log into the database instance as a user who has been granted the `DV_ACCTMGR` role.

   For example:

   ```
   sqlplus bea_dvacctmgr
   Enter password: password
   ```

   In a multitenant environment, you must log in to the appropriate pluggable database (PDB). For example:

   ```
   sqlplus bea_dvacctmgr@hrpdb
   Enter password: password
   ```

   To find the available PDBs, query the `DBA_PDBS` data dictionary view. To check the current PDB, run the `show con_name` command.

2. Create the following users and grant them the `CREATE SESSION` privilege.

   ```
   GRANT CREATE SESSION TO patch_boss IDENTIFIED BY password;
   GRANT CREATE SESSION TO patch_user IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

3. Connect as user `SYS` with the `SYSDBA` administrative privilege.

   ```
   CONNECT SYS AS SYSDBA -- Or, CONNECT SYS@hrpdb AS SYSDBA
   Enter password: password
   ```

4. Grant the following privileges to the `DV_OWNER` or `DV_ADMIN` user.

   For example:

   ```
   GRANT CREATE PROCEDURE TO leo_dvowner;
   GRANT SELECT ON V_$SESSION TO leo_dvowner;
   ```

The V_$SESSION table is the underlying table for the V$SESSION dynamic view.

In a real-world scenario, you also would log in as the DV_OWNER user and grant the DV_PATCH_ADMIN role to user patch_user (but not to patch_boss). But because you are not really going to perform a database patch upgrade in this tutorial, you do not need to grant this role to user patch_user.

Step 2: Create a Function to Check if User patch_boss Is Logged In

The behavior of the Database Vault settings will be determined by the function.

The function that you must create, check_boss_logged_in, does just that: When user patch_user tries to log into the database instance, it checks if user patch_boss is already logged in by querying the V$SESSION data dictionary view.

1. Connect as a user who has been granted the DV_OWNER or DV_ADMIN role.

   For example:

   ```
   CONNECT leo_dvowner -- Or, CONNECT leo_dvowner@hrpdb
   Enter password: password
   ```

2. Create the check_boss_logged_in function as follows:

   ```
   CREATE OR REPLACE FUNCTION check_boss_logged_in
   return varchar2
   authid definer as

   v_session_number number := 0;
   v_allow varchar2(10)    := 'TRUE';
   v_deny varchar2(10)     := 'FALSE';

   BEGIN
     SELECT COUNT(*) INTO v_session_number
     FROM SYS.V_$SESSION
     WHERE USERNAME = 'PATCH_BOSS'; -- Enter the user name in capital letters.

    IF v_session_number > 0
     THEN RETURN v_allow;
    ELSE
     RETURN v_deny;
    END IF;
   END check_boss_logged_in;
   /
   ```

3. Grant the EXECUTE privilege on the check_boss_logged_in function to the DVSYS schema.

   ```
   GRANT EXECUTE ON check_boss_logged_in to DVSYS;
   ```

Step 3: Create Rules, a Rule Set, and a Command Rule to Control User Access

Next, you must create two rules, a rule set to which you will add them, and a command rule.

The rule set triggers the check_boss_logged_in function when user patch_user tries to logs in to the database.

1. Connect as a user who has been granted the DV_OWNER or DV_ADMIN role.

For example:

```
CONNECT leo_dvowner -- Or, CONNECT leo_dvowner@hrpdb
Enter password: password
```

2. Create the Check if Boss Is Logged In rule, which checks that the `patch_user` user is logged in to the database. In the definition, replace `leo_dvowner` with the name of the `DVOWNER` or `DV_ADMIN` user who created the `check_boss_logged_in` function.

   If the `check_boss_logged_in` function returns `TRUE` (that is, `patch_boss` is logged in to another session), then `patch_user` can log in.

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check if Boss Is Logged In',
   rule_expr => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''PATCH_USER'' and
leo_dvowner.check_boss_logged_in =  ''TRUE'' ');
END;
/
```

   Enter the user name, `PATCH_USER`, in upper-case letters, which is how the `SESSION_USER` parameter stores it.

3. Create the Allow Connect for Other Database Users rule, which ensures that the user logged in (`patch_user`) is not user `patch_boss`. It also enables all other valid users to log in.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Allow Connect for Other Database Users',
  rule_expr => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') != ''PATCH_USER''');
END;
/
COMMIT;
```

4. Create the Dual Connect for Boss and Patch rule set, and then add the two rules to it.

```
BEGIN
    DBMS_MACADM.CREATE_RULE_SET(
     rule_set_name     => 'Dual Connect for Boss and Patch',
     description       => 'Checks if both boss and patch users are logged in.',
     enabled           => DBMS_MACUTL.G_YES,
     eval_options      => 2,
     audit_options     => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
     fail_options      => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
     fail_message      =>'',
     fail_code         => NULL,
     handler_options   => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
     handler           => ''
     );
END;
/

BEGIN
    DBMS_MACADM.ADD_RULE_TO_RULE_SET(
     rule_set_name     => 'Dual Connect for Boss and Patch',
     rule_name         => 'Check if Boss Is Logged In'
   );
END;
/

BEGIN
```

```
    DBMS_MACADM.ADD_RULE_TO_RULE_SET(
     rule_set_name     => 'Dual Connect for Boss and Patch',
     rule_name         => 'Allow Connect for Other Database Users'
   );
END;
/
```

5. Create the following CONNECT command rule, which permits user `patch_user` to connect to the database only if `patch_boss` is already logged in.

```
BEGIN
   DBMS_MACADM.CREATE_COMMAND_RULE(
    command            => 'CONNECT',
    rule_set_name      => 'Dual Connect for Boss and Patch',
    object_owner       => '%',
    object_name        => '%',
    enabled            => DBMS_MACUTL.G_YES);
END;
/
COMMIT;
```

Step 4: Test the Users' Access

After the rules have been created, they are ready to be tested.

1. Exit SQL*Plus.

```
EXIT
```

2. Create a second shell, for example:

```
xterm &
```

3. In the first shell, try to log in as user `patch_user`.

```
sqlplus patch_user -- Or, sqlplus patch_user@hrpdb
Enter password: password

ERROR:
ORA-47400: Command Rule violation for CONNECT on LOGON

Enter user-name:
```

User `patch_user` cannot log in until user `patch_boss` is already logged in. (Do not try the `Enter user-name` prompt yet.)

4. In the second shell and then log in as user `patch_boss`.

```
sqlplus patch_boss -- Or, sqlplus patch_boss@hrpdb
Enter password: password
Connected.
```

User `patch_boss` can log in.

5. Go back to the first shell, and then try logging in as user `patch_user` again.

```
Enter user_name: patch_user
Enter password: password
```

This time, user `patch_user` is deemed a valid user, so now he can log in.

Step 5: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. In the session for the user `patch_boss`, exit SQL*Plus and then close the shell.

   ```
   EXIT
   ```

2. In the first shell, connect the `DV_ACCTMGR` user and remove the users you created.

   ```
   CONNECT bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
   Enter password: password

   DROP USER patch_boss;
   DROP USER patch_user;
   ```

3. Connect as a user `SYS` with the `SYSDBA` administrative privilege and revoke the privileges that you had granted to the `DV_OWNER` or `DV_ADMIN` user.

   ```
   CONNECT SYS AS SYSDBA -- Or, CONNECT SYS@hrpdb AS SYSDBA
   Enter password: password

   REVOKE CREATE PROCEDURE FROM leo_dvowner;
   REVOKE SELECT ON V_$SESSION FROM leo_dvowner;
   ```

4. Connect as the `DV_OWNER` or `DV_ADMIN` user and drop the rules, rule set, and command rule, in the order shown.

   ```
   CONNECT leo_dvowner -- Or, CONNECT leo_dvowner@hrpdb
   Enter password: password

   DROP FUNCTION check_boss_logged_in;
   EXEC DBMS_MACADM.DELETE_COMMAND_RULE('CONNECT', '%', '%');
   EXEC DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('Dual Connect for Boss and Patch',
   'Check if Boss Is Logged In');
   EXEC DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('Dual Connect for Boss and Patch',
   'Allow Connect for Other Database Users');
   EXEC DBMS_MACADM.DELETE_RULE('Check if Boss Is Logged In');
   EXEC DBMS_MACADM.DELETE_RULE('Allow Connect for Other Database Users');
   EXEC DBMS_MACADM.DELETE_RULE_SET('Dual Connect for Boss and Patch');
   COMMIT;
   ```

Guidelines for Designing Rule Sets

Oracle provides guidelines for designing rule sets.

- You can share rules among multiple rule sets. This lets you develop a library of reusable rule expressions. Oracle recommends that you design such rules to be discrete, single-purpose expressions.
- You can design a rule set so that its evaluation is static, that is, it is evaluated only once during a user session. Alternatively, it can be evaluated each time the rule set is accessed. If the rule set is evaluated only once, then the evaluated value is reused throughout the user session each time the rule set is accessed. Using static evaluation is useful in cases where the rule set must be accessed multiple times but the conditions on which the rule set depend do not change during that session. An example would be a SELECT command rule associated with a rule set when the same `SELECT` statement occurs multiple times and if the evaluated

value is acceptable to use again, rather than evaluating the rule set each time the SELECT occurs.

To control the static evaluation of the rule set, set the is_static parameter of the CREATE_RULE_SET or UPDATE_RULE_SET procedures of the DBMS_MACADM PL/SQL package. See DBMS_MACADM Rule Set Procedures for more information.

- Use Oracle Database Vault factors in your rule expressions to provide reusability and trust in the values used by your rule expressions. Factors can provide contextual information to use in your rules expressions.
- You can use custom event handlers to extend Oracle Database Vault security policies to integrate external systems for error handling or alerting. Using Oracle utility packages such as UTL_TCP, UTL_HTTP, UTL_MAIL, UTL_SMTP, or DBMS_AQ can help you to achieve this type of integration.
- Test rule sets thoroughly for various accounts and scenarios either on a test database or on a test realm or command rule for nonsensitive data before you apply them to realms and command rules that protect sensitive data. You can test rule expressions directly with the following SQL statement:

  ```
  SQL> SELECT SYSDATE from DUAL where rule expression
  ```

- You can nest rule expressions inside a single rule. This helps to achieve more complex situations where you would need a logical AND for a subset of rules and a logical OR with the rest of the rules. See the definition for the Is Corporate Network During Maintenance rule set under Tutorial: Creating an Email Alert for Security Violations for an example.
- You cannot use invoker's rights procedures with rule expressions. Only use definer's rights procedures with rule expressions.

How Rule Sets Affect Performance

The number and complexity of rules can slow database performance.

Rule sets govern the performance for execution of certain operations. For example, if you have a very large number of rules in a rule set governing a SELECT statement, performance could degrade significantly.

If you have rule sets that require many rules, performance improves if you move all the rules to logic defined in a single PL/SQL standalone or package function. However, if a rule is used by other rule sets, there is little performance effect on your system.

If possible, consider setting the rule set to use static evaluation, assuming this is compatible with the associated command rule's usage. See Guidelines for Designing Rule Sets for more information.

You can check system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Cloud Control, which is installed by default with Oracle Database), Automatic Workload Repository (AWR), and TKPROF.

See Also:

- *Oracle Database Performance Tuning Guide* to learn how to monitor database performance
- *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements

Rule Set and Rule Related Reports and Data Dictionary Views

Oracle Database Vault provides reports and data dictionary views that are useful for analyzing rule sets and the rules within them.

Table 6-2 lists the Oracle Database Vault reports. See Oracle Database Vault Reports for information about how to run these reports.

Table 6-2 Reports Related to Rule Sets

Report	Description
Rule Set Configuration Issues Report	Lists rule sets that have no rules defined or enabled
Secure Application Configuration Issues Report	Lists secure application roles that have incomplete or disabled rule sets
Command Rule Configuration Issues Report	Lists rule sets that are incomplete or disabled

Table 6-3 lists data dictionary views that provide information about existing rules and rule sets.

Table 6-3 Data Dictionary Views Used for Rules and Rule Sets

Data Dictionary View	Description
DBA_DV_RULE View	Lists the rules that have been defined
DBA_DV_RULE_SET View	Lists the rule sets that have been created
DBA_DV_RULE_SET_RULE View	Lists rules that are associated with existing rule sets

7

Configuring Command Rules

You can create command rules or use the default command rules to protect DDL and DML statements.

- What Are Command Rules?
  A command rule applies Oracle Database Vault protections with an Oracle Database SQL statement, such as `ALTER SESSION`.
- Default Command Rules
  Oracle Database Vault provides default command rules, based on commonly used SQL statements.
- SQL Statements That Can Be Protected by Command Rules
  You can protect a large number of SQL statements by using command rules.
- Creating a Command Rule
  You can create a command rule in Oracle Database Vault Administrator.
- Modifying the Enablement Status of a Command Rule
  You can enable or disable a command rule in Oracle Database Vault Administrator.
- Deleting a Command Rule
  Before you delete a command rule, you can locate the various references to it by querying the command rule-related Oracle Database Vault views.
- How Command Rules Work
  Command rules follow a set of steps to check their associated components.
- Tutorial: Using a Command Rule to Control Table Creations by a User
  In this tutorial, you create a simple local command rule to control whether users can create tables in the `SCOTT` schema.
- Guidelines for Designing Command Rules
  Oracle provides guidelines for designing command rules.
- How Command Rules Affect Performance
  The performance of a command rule depends on the complexity of the rules in the rule set associated with the command rule.
- Command Rule Related Reports and Data Dictionary View
  Oracle Database Vault provides reports and a data dictionary view that are useful for analyzing command rules.

What Are Command Rules?

A command rule applies Oracle Database Vault protections with an Oracle Database SQL statement, such as `ALTER SESSION`.

- About Command Rules
  A command rule protects Oracle Database SQL statements that affect one or more database objects.

- Command Rules in a Multitenant Environment
  In a multitenant environment, you can create common and local command rules in either the CDB root or the application root.
- Types of Command Rules
  In addition to command rules for many SQL statements, you can create command rules specifically for the `CONNECT`, `ALTER SYSTEM`, and `ALTER SESSION` SQL statements.

About Command Rules

A command rule protects Oracle Database SQL statements that affect one or more database objects.

These statements can include `SELECT`, `ALTER SYSTEM`, database definition language (DDL), and data manipulation language (DML) statements.

To customize and enforce the command rule, you associate it with a rule set, which is a collection of one or more rules. The command rule is enforced at run time. Command rules affect anyone who tries to use the SQL statements it protects, regardless of the realm in which the object exists.

You can use command rules to protect a wide range of SQL statements, in addition to basic Oracle Database DDL and DML statements. For example, you can protect statements that are used with Oracle Flashback Technology.

A command rule has the following attributes, in addition to associating a command rule to a command:

- SQL statement the command rule protects
- Owner of the object the command rule affects
- Database object the command rule affects
- Whether the command rule is enabled
- An associated rule set

Command rules can be categorized as follows:

- **Command rules that have a system-wide scope.** With this type, in most cases, you can only create one command rule for each database instance.
- **Command rules that are schema specific.** An example of a schema-specific command rule is a command rule for the `DROP TABLE` statement. You can create only one CONNECT command rule for each schema.
- **Command rules that are object specific.** An example is creating a command rule for the `DROP TABLE` statement with a specific table included in the command rule definition.

When a user executes a statement affected by a command rule, Oracle Database Vault checks the realm authorization first. If it finds no realm violation and if the associated command rules are enabled, then Database Vault evaluates the associated rule sets. If all the rule sets evaluate to `TRUE`, then the statement is authorized for further processing. If any of the rule sets evaluate to `FALSE`, then the statement is not allowed to be executed and a command rule violation is raised.

You can define a command rule that uses factors for the `CONNECT` event to permit or deny sessions after the usual steps–user authentication process, factor initialization,

and Oracle Label Security integration–are complete. For example, you can configure a command rule that allows DDL statements such as `CREATE TABLE`, `DROP TABLE`, and `ALTER TABLE` in the `BIZAPP` schema to be authorized after business hours, but not during business hours.

You can run reports on the command rules that you create in Oracle Database Vault.

You cannot create command rules that block `SYS` from executing `SYS`-owned procedures.

Related Topics

- Oracle Database Vault Command Rule APIs
  The `DBMS_MACADM` PL/SQL package provides procedures for configuring command rules. .
- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.
- SQL Statements That Can Be Protected by Command Rules
  You can protect a large number of SQL statements by using command rules.

Command Rules in a Multitenant Environment

In a multitenant environment, you can create common and local command rules in either the CDB root or the application root.

Common command rules can be associated only with common realms, rule sets, and rules. Local command rules can be associated only with local realm, rule sets, and rules.

To apply these command rules to the entire multitenant environment, you must execute the command rule procedures from the CDB root or application root as a common user who has been granted the `DVADM` or `DVOWNER` role. A common command rule that is created in the CDB root will be applied to all PDBs in that CDB environment. A common command rule that is created in the application root will only be applied to the PDBs that are associated with this application root. To propagate the command rule to the PDBs that are associated with the CDB root or application root, you must synchronize the PDB. For example, to synchronize an application root called `saas_sales_app` to its application PDBs:

```
ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;
```

To synchronize a common command rule in the CDB root to a PDB:

```
ALTER PLUGGABLE DATABASE APPLICATION APP$CDB$SYSTEM SYNC;
```

You can check a user's roles by querying the `USER_ROLE_PRIVS` data dictionary view. To find information about command rules, query the `DBA_DV_COMMAND_RULE` data dictionary view.

Types of Command Rules

In addition to command rules for many SQL statements, you can create command rules specifically for the `CONNECT`, `ALTER SYSTEM`, and `ALTER SESSION` SQL statements.

- CONNECT Command Rule
  The `DBMS_MACADM.CREATE_CONNECT_CMD_RULE` procedure creates a user-specific CONNECT command rule.
- ALTER SESSION and ALTER SYSTEM Command Rules
  You can create different kinds of `ALTER SESSION` and `ALTER SYSTEM` command rules that provide fine-grained control for these SQL statements.

CONNECT Command Rule

The `DBMS_MACADM.CREATE_CONNECT_CMD_RULE` procedure creates a user-specific CONNECT command rule.

This type of command rule specifies a user, an associated rule set, an enablement status, and for a multitenant environment, where to execute the CONNECT command rule. You can enable or disable the CONNECT command rule, or you can set it to use simulation mode. In simulation mode, violations to the command rule are logged in a designated log table with sufficient information to describe the error, such as the user name or SQL statement used.

In a multitenant environment, you can create the CONNECT command rule in either the application root or in a specific PDB. The associated rule set must be consistent with the CONNECT command rule: if the CONNECT command rule is in the application root, then the rule set and rules must also be in the application root. You run the CONNECT command rule procedures from the CDB root as a common user. If the CONNECT command rule is local to a pluggable database (PDB), then you must run the CONNECT command rule creation command in that PDB, and the rule set and rules must be local.

The following example shows a CONNECT command rule definition that creates a local, enabled CONNECT command rule for the `HR` user. The rule set that is associated with this command rule is local to the current PDB.

```
BEGIN
DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
 rule_set_name   => 'Enabled',
 user_name       => 'HR',
 enabled         => DBMS_MACUTL.G_YES,
 scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

Related Topics

- CREATE_COMMAND_RULE Procedure
  The `CREATE_COMMAND_RULE` procedure creates a command rule and associates it with a rule set.
- Using Simulation Mode for Logging Realm and Command Rule Activities
  Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

ALTER SESSION and ALTER SYSTEM Command Rules

You can create different kinds of `ALTER SESSION` and `ALTER SYSTEM` command rules that provide fine-grained control for these SQL statements.

The procedures to create these types of command rules are as follows:

- `DBMS_MACADM.CREATE_COMMAND_RULE` creates ALTER SESSION and ALTER SYSTEM command rules that use clauses from the corresponding SQL statement, such as `ADVISE`, `CLOSE DATABASE LINK`, `COMMIT IN PROCEDURE`, and `SET` for `ALTER SESSION`, or `ARCHIVE_LOG`, `CHECK DATAFILES`, `CHECKPOINT`, and `SET` for `ALTER SYSTEM`.
- `DBMS_MACADM.CREATE_SESSION_EVENT` creates a command rule that is specific to the `ALTER SESSION SET EVENTS` SQL statement
- `DBMS_MACADM_CREATE_SYSTEM_EVENT` creates a command rule that is specific to the `ALTER SYSTEM SET EVENTS` SQL statement.

To create these command rules, you use the appropriate Database Vault procedure to specify the clause and if applicable, the parameter of the clause, in the creation statement. If the ALTER SESSION or ALTER SYSTEM command rule use the `SET EVENTS` setting, then you can use special parameters to specify events, components, and actions.

For example, for an ALTER SYSTEM command rule, you could specify the `SECURITY` clause and its `RESTRICTED SESSION` parameter from the `ALTER SYSTEM` SQL statement. To specify whether `RESTRICTED SESSION` is `TRUE` or `FALSE`, you must create a Database Vault rule and rule set, which can test for the validity of this sequence number.

To understand how this concept works, first create the following rule and rule set, which are designed to check if the `RESTRICTED SESSION` parameter is set to `TRUE`:

```
EXEC DBMS_MACADM.CREATE_RULE('RESTRICTED SESSION TRUE', 'UPPER(PARAMETER_VALUE) =
''TRUE''');

BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Check RESTRICTED SESSION for TRUE',
  description      => 'Checks if restricted session is true',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL +
DBMS_MACUTL.G_RULESET_AUDIT_SUCCESS,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
  fail_message     => 'RESTRICTED SESSION is not TRUE',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler          => '',
  is_static        => false);
END;
/
EXEC DBMS_MACADM.ADD_RULE_TO_RULE_SET(Check RESTRICTED SESSION for TRUE',
'RESTRICTED SESSION TRUE');
```

With the rule and rule set in place, you are ready to create an ALTER SYSTEM command rule that will check if the `RESTRICTED SESSION` parameter:

```
BEGIN
DBMS_MACADM.CREATE_COMMAND_RULE(
 command         => 'ALTER SYSTEM',
 rule_set_name   => 'Check RESTRICTED SESSION for TRUE',
 object_owner    => '%',
 object_name     => '%',
 enabled         => DBMS_MACUTL.G_YES,
 clause_name     => 'SECURITY',
 parameter_name  => 'RESTRICTED SESSION',
 scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
```

```
END;
/
```

In this example:

- `rule_set_name` checks whether `RESTRICTED SESSION` is set to `TRUE` or `FALSE`. In a multienant environment, you must create the rule set and rule in the same location as the command rule: either in the application root or locally in a PDB.
- `object_owner` and `object_name` must always be set to `%` for this kind of `ALTER SESSION` or `ALTER SYSTEM` command rule.
- `enabled` enables you to enable or disable the command rule, or to use simulation mode to log violations to the command rule to a designated log table. The log data describes the error, such as the user name or SQL statement used.
- `clause_name` specifies the `SECURITY` clause of the `ALTER SYSTEM` SQL statement
- `parameter_name` specifies the `RESTRICTED SESSION` parameter from the `SECURITY` clause
- `scope` sets the command rule to be local to the current PDB. The associated rule set and rule must also be local to the current PDB. If you want to create the command rule in the application root, then as a common user, you would set `scope` to `DBMS_MACUTL.G_SCOPE_COMMON` and run the procedure (and its accompanying rule set and rule creation procedures) from the application root.

See Also:

- CREATE_COMMAND_RULE Procedure about the `DBMS_MACADM.CREATE_COMMAND_RULE` procedure
- CREATE_SESSION_EVENT_CMD_RULE Procedure about the `DVS.DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE` procedure
- CREATE_SYSTEM_EVENT_CMD_RULE Procedure for more information about the `DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE` procedure
- DBA_DV_COMMAND_RULE View for information about the `DBA_DV_COMMAND_RULE` data dictionary view
- *Oracle Database SQL Language Reference* for information about the `ALTER SESSION` SQL statement
- *Oracle Database SQL Language Reference* for information about the `ALTER SYSTEM` SQL statement

Default Command Rules

Oracle Database Vault provides default command rules, based on commonly used SQL statements.

Table 7-1 lists the default Database Vault command rules.

Table 7-1 Default Command Rules

SQL Statement	Rule Set Name
CREATE USER	Can Maintain Accounts/Profiles
ALTER USER	Can Maintain Own Account
DROP USER	Can Maintain Accounts/Profiles
CREATE PROFILE	Can Maintain Accounts/Profiles
ALTER PROFILE	Can Maintain Accounts/Profiles
DROP PROFILE	Can Maintain Accounts/Profiles
ALTER SYSTEM	Allow Fine Grained Control of System Parameters
CHANGE PASSWORD	Can Maintain Own Account[1]

[1] The actual SQL statement that the Can Maintain Own Account rule refers to is PASSWORD.

The following set of command rules helps you to achieve separation of duty for user management:

- ALTER PROFILE
- ALTER USER
- CREATE PROFILE
- CREATE USER
- DROP PROFILE
- DROP USER

To grant a user the ability to use these commands, you can grant the user the role that the rule set checks. For example, the CREATE USER command rule ensures that a user who tries to run a CREATE USER statement has the DV_ACCTMGR role.

> **Note:**
>
> To find information about the default command rules, query the DBA_DV_COMMAND_RULE data dictionary view.

SQL Statements That Can Be Protected by Command Rules

You can protect a large number of SQL statements by using command rules.

The SQL statements that you can protect are as follows:

SQL Statements A-A	SQL Statements A-C	SQL Statements C-U
ALTER CLUSTER	ANALYZE TABLE	CREATE TABLE

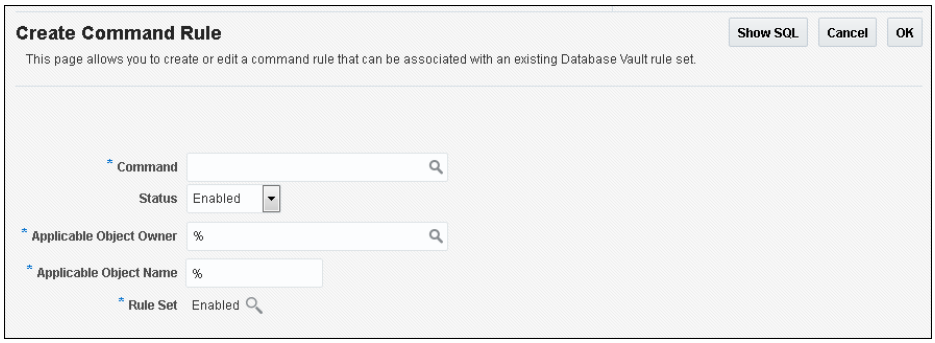
SQL Statements A-A	SQL Statements A-C	SQL Statements C-U
ALTER DIMENSION	ASSOCIATE STATISTICS	CREATE TABLESPACE
ALTER FLASHBACK ARCHIVE	AUDIT	CREATE TRIGGER
ALTER FUNCTION	CHANGE PASSWORD	CREATE TYPE
ALTER INDEX	COMMENT	CREATE TYPE BODY
ALTER INDEXTYPE	CONNECT	CREATE VIEW
ALTER JAVA	CREATE EDITION	DELETE
ALTER LIBRARY	CREATE FLASHBACK ARCHIVE	DISASSOCIATE STATISTICS
ALTER OPERATOR	CREATE USER	DROP CLUSTER
ALTER OUTLINE	CREATE CLUSTER	DROP CONTEXT
ALTER MATERIALIZED VIEW	CREATE CONTEXT	DROP DATABASE LINK
ALTER MATERIALIZED VIEW LOG	CREATE DATABASE LINK	DROP EDITION
ALTER PACKAGE	CREATE DIMENSION	DROP DIMENSION
ALTER PACKAGE BODY	CREATE DIRECTORY	DROP DIRECTORY
ALTER PLUGGABLE DATABASE	CREATE FUNCTION	DROP FLASHBACK ARCHIVE
ALTER PROCEDURE	CREATE INDEX	DROP FUNCTION
ALTER PROFILE	CREATE INDEXTYPE	FLASHBACK TABLE
ALTER RESOURCE COST	CREATE JAVA	EXECUTE
ALTER ROLE	CREATE LIBRARY	GRANT
ALTER ROLLBACK SEGMENT	CREATE OPERATOR	INSERT
ALTER SEQUENCE	CREATE OUTLINE	NOAUDIT
ALTER SESSION	CREATE PACKAGE	PURGE DBA_RECYCLEBIN
ALTER SYNONYM	CREATE PACKAGE BODY	PURGE INDEX
ALTER SYSTEM	CREATE PLUGGABLE DATABASE	RENAME
ALTER TABLE	CREATE PROCEDURE	PURGE RECYCLEBIN
ALTER TABLESPACE	CREATE PROFILE	PURGE TABLE
ALTER TRIGGER	CREATE ROLE	PURGE TABLESPACE
ALTER TYPE	CREATE ROLLBACK SEGMENT	REVOKE
ALTER TYPE BODY	CREATE SCHEMA	SELECT
ALTER USER	CREATE SEQUENCE	TRUNCATE CLUSTER
ALTER VIEW	CREATE MATERIALIZED VIEW	TRUNCATE TABLE
ANALYZE CLUSTER	CREATE MATERIALIZED VIEW LOG	UPDATE
ANALYZE INDEX	CREATE SYNONYM	-

> **See Also:**
>
> Command Rules in a Multitenant Environment for information about using `CREATE PLUGGABLE DATABASE`, `ALTER PLUGGABLE DATABASE`, and `DROP PLUGGABLE DATABASE` in a multitenant container database (CDB)

Creating a Command Rule

You can create a command rule in Oracle Database Vault Administrator.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Command Rules**.
3. In the Command Rules page:
   - To create a new command rule, click **Create** to display the Create Command Rule page.



4. In the Create Command Rule page, enter the following settings:
   - **Command:** Select the SQL statement or operation for which you want to create a command rule. This attribute is mandatory.
   - **Status:** Select either **Enabled**, **Disabled**, or **Simulation**, which will apply to the command rule during run time. This attribute is mandatory.
   - **Applicable Object Owner:** From the list, select the owner of the object the command rule affects. You can use wildcard character `%` to select all owners. (However, you cannot use wildcard characters with text, such as `EM%` to select all owners whose names begin in `EM`.) This attribute is mandatory for all SQL statements that operate on objects within a specific schema. See SQL Statements That Can Be Protected by Command Rules for a list of supported SQL statements.

     Note that the `SELECT`, `INSERT`, `UPDATE`, `DELETE`, and `EXECUTE` statements are not allowed for a selection of all (%) or the `SYS` and `DVSYS` schemas.

- **Applicable Object Name:** Enter the name of the database object that the command rule affects, or specify % to select all database objects. This attribute is mandatory, if you selected an object owner from the Object Owner list.

  You can run Oracle Database Vault reports on objects that the command rule affects. See the Command Rule Related Reports and Data Dictionary View for more information.

- **Rule Set:** From the list, select the rule set that you want to associate with the command rule. This attribute is mandatory.

  If the rule set evaluates to true, then the SQL statement succeeds. If it evaluates to false, the statement fails, and then Oracle Database Vault raises a command rule violation. (You can track rule violations by using the Command Rule Configuration Issues Report, discussed in Oracle Database Vault Reports.) Any auditing and custom event handling associated with the rule set occurs as a part of the command rule processing.

  See Configuring Rule Sets , for more information about rule sets.

5. Click **OK**.

Related Topics

- Propagating Oracle Database Vault Configurations to Other Databases
  You can propagate Database Vault configurations (such as a realm configuration) to other Database Vault-protected databases.

Modifying the Enablement Status of a Command Rule

You can enable or disable a command rule in Oracle Database Vault Administrator.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Command Rules**.
3. In the Command Rules page, select the command rule that you want to enable or disable, and then select **Edit**.
4. In the Edit Command Rule page, select the status that you want from the **Status** menu:
   - **Enabled**
   - **Disabled**
   - **Simulation**
5. Click **OK**.

Deleting a Command Rule

Before you delete a command rule, you can locate the various references to it by querying the command rule-related Oracle Database Vault views.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Oracle Database Vault Administration page, select **Command Rules**.
3. In the Command Rules page, select the command rule that you want to remove.
4. Click **Delete**.
5. In the Confirmation window, click **Yes**.

Related Topics

- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

How Command Rules Work

Command rules follow a set of steps to check their associated components.

How Realms Work describes what happens when a database account issues a `SELECT`, DDL, or DML statement that affects objects within a realm.

The following actions take place when `SELECT`, DDL, or DML statement is issued:

1. Oracle Database Vault queries all the command rules that need to be applied.

   For `SELECT`, DDL, and DML statements, multiple command rules may apply because the object owner and object name support wildcard notation.

   You can associate rule sets with both command rules and realm authorizations. Oracle Database Vault evaluates the realm authorization rule set first, and then it evaluates the rule sets that apply to the command type being evaluated.
2. For each command rule that applies, Oracle Database Vault evaluates its associated rule set.
3. If the associated rule set of any of the applicable command rules returns false or errors, Oracle Database Vault prevents the command from executing. Otherwise, the command is authorized for further processing. The configuration of the rule set with respect to auditing and event handlers dictates the auditing or custom processing that occurs.

   Command rules override object privileges. That is, even the owner of an object cannot access the object if the object is protected by a command rule. You can disable either a command rule or the rule set of a command. If you disable a command rule, then the command rule does not perform the check it is designed to handle. If you disable a rule set, then the rule set always evaluates to `TRUE`. However, if you want to disable a command rule for a particular command, then you should disable the command rule because the rule set may be associated with other command rules or realm authorizations.

Tutorial: Using a Command Rule to Control Table Creations by a User

In this tutorial, you create a simple local command rule to control whether users can create tables in the SCOTT schema.

- Step 1: Create a Table
  First, user SCOTT must create a table.
- Step 2: Create a Command Rule
  After the table has been created in the SCOTT schema, you can create a command rule.
- Step 3: Test the Command Rule
  Next, you are ready to test the CREATE TABLE local command rule.
- Step 4: Remove the Components for this Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

Step 1: Create a Table

First, user SCOTT must create a table.

1. Log into the database instance as user SCOTT.

   ```
   sqlplus scott
   Enter password: password
   ```

   In a multitenant environment, you must log in to the appropriate PDB. For example:

   ```
   sqlplus scott@hrpdb
   Enter password: password
   ```

   To find the available pluggable databases (PDBs), query the DBA_PDBS data dictionary view. To check the current PDB, run the show con_name command.

   If the SCOTT account is locked and expired, then log in as the Database Vault Account Manager and unlock SCOTT and create a new password. For example:

   ```
   sqlplus bea_dvacctmgr --Or, sqlplus bea_dvacctmgr@hrpdb
   Enter password: password

   ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

   ```
   CONNECT SCOTT --Or, sqlplus SCOTT@hrpdb
   Enter password: password
   ```

2. As user SCOTT, create a table.

   ```
   CREATE TABLE t1 (num NUMBER);
   ```

3. Now drop the table.

   ```
   DROP TABLE t1;
   ```

At this stage, user SCOTT can create and drop tables. Do not exit SQL*Plus yet, and remain connected as SCOTT. You must use it later on when SCOTT tries to create another table.

Step 2: Create a Command Rule

After the table has been created in the SCOTT schema, you can create a command rule.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the DV_OWNER or DV_ADMIN role and the SELECT ANY DICTIONARY privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Oracle Database Vault Administrator Administration page, click **Command Rules**.

   The Command Rules page appears.
3. Click **Create**.

   The Create Command Rule page appears.
4. Enter the following settings:
   - **Command:** Select **CREATE TABLE**
   - **Status:** Set to **Enabled** so that the command rule is active.
   - **Applicable Object Owner:** Select **SCOTT**.
   - **Applicable Object Name:** Set to **%** so that it applies to all objects in the SCOTT schema.
   - **Rule Set:** Select **Disabled** so that no one can create tables in the SCOTT schema.
5. Click **OK**.

   Do not exit Database Vault Administrator

Command rules take effect immediately. Right away, user SCOTT is prevented from creating tables, even though he is still in the same user session he was in a moment ago, before you created the CREATE TABLE command rule.

Step 3: Test the Command Rule

Next, you are ready to test the CREATE TABLE local command rule.

1. In SQL*Plus, ensure that you are logged on as user SCOTT.

   ```
   CONNECT SCOTT --Or, CONNECT SCOTT@hrpdb
   Enter password: password
   ```
2. Try to create a table.

   ```
   CREATE TABLE t1 (num NUMBER);
   ```

   The following output should appear:

   ```
   ORA-47400: Command Rule violation for create table on SCOTT.T1
   ```

As you can see, SCOTT is no longer allowed to create tables, even in his own schema.

3. In Oracle Database Vault Administrator, do the following:
   a. In the Command Rules page, select the CREATE TABLE command rule and then click **Edit**.
   b. In the Edit Command Rule page, select **Enabled** from the **Rule Set** list.
   c. Click **OK**.
4. In SQL*Plus, as user SCOTT, try creating the table again.

```
CREATE TABLE t1 (num NUMBER);

Table created.
```

Now that the CREATE TABLE command rule is set to Enabled, user SCOTT is once again permitted to create tables. (Do not exit SQL*Plus.)

Step 4: Remove the Components for this Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. In Oracle Database Vault Administrator, remove the CREATE TABLE command rule as follows:
   a. Return to the Command Rules page.
   b. Select the CREATE TABLE local command rule and then click **Delete**.
   c. In the Confirmation window, click **Yes**.
2. Log into the database instance as user SCOTT and remove the t1 table.

```
DROP TABLE t1;
```

3. If you no longer need the SCOTT account to be available, then connect as the Database Vault Account Manager and enter the following ALTER USER statement:

```
CONNECT bea_dvacctmgr --Or, CONNECT bea_dvacctmgr@hrpdb
Enter password: password

ALTER USER SCOTT ACCOUNT LOCK PASSWORD EXPIRE;
```

Guidelines for Designing Command Rules

Oracle provides guidelines for designing command rules.

-
- Create finer-grained command rules, because they are far easier to maintain.

  For example, if you want to prevent SELECT statements from occurring on specific schema objects, then design multiple command rules to stop the SELECT statements on those specific schema objects, rather than creating a general command rule to prevent SELECT statements in the schema level.
- When designing rules for the CONNECT event, be careful to include logic that does not inadvertently lock out any required user connections. If any account has been locked out accidentally, ask a user who has been granted the DV_ADMIN or

DV_OWNER role to log in and correct the rule that is causing the lock-out problem. The CONNECT command rule does not apply to users with the DV_OWNER and DV_ADMIN roles. This prevents improperly configured CONNECT command rules from causing a complete lock-out.

If the account has been locked out, you can disable Oracle Database Vault, correct the rule that is causing the lock-out problem, and then reenable Oracle Database Vault. Even when Oracle Database Vault is disabled, you still can use Database Vault Administrator and the Database Vault PL/SQL packages. See Disabling and Enabling Oracle Database Vault, for instructions on disabling and reenabling Database Vault.

- Sometimes you must temporarily relax an enabled command rule for an administrative task. Rather than disabling the command rule, have the Security Manager (the account with the DV_ADMIN or DV_OWNER role) log in, set the rule set to **Enabled**, turn on **Auditing on Success or Failure** for the default rule set named Enabled, and then set the command rule back to its original rule set when the task is complete. (Be aware that in a unified auditing environment, this setting does not work. Instead, you must create a unified audit policy. *Oracle Database Security Guide* describes how to create unified audit policies for Database Vault.)
- When designing command rules, be careful to consider automated processes such as backup where these procedures may be inadvertently disabled. You can account for these tasks by creating rules that allow the command when a series of Oracle Database Vault factors is known to be true (for example, the program being used), and the account being used or the computer or network on which the client program is running.
- You can test the development phase of a command rule by using simulation mode, which enables the command rule but writes detailed information about it to a log file.

Related Topics

- Using Simulation Mode for Logging Realm and Command Rule Activities
  Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

How Command Rules Affect Performance

The performance of a command rule depends on the complexity of the rules in the rule set associated with the command rule.

For example, suppose a rule set invokes a PL/SQL function that takes 5 seconds to run. In this case, a command rule that uses that rule set would take 5 seconds to grant access for the command statement to run.

You can check the system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Cloud Control, which is installed by default with Oracle Database), Automatic Workload Repository (AWR), and TKPROF.

See Also:

- *Oracle Database Performance Tuning Guide* to learn how to monitor database performance
- *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements

Command Rule Related Reports and Data Dictionary View

Oracle Database Vault provides reports and a data dictionary view that are useful for analyzing command rules.

Table 7-2 lists the Oracle Database Vault report. See Oracle Database Vault Reports, for information about how to run these reports.

Table 7-2 Reports Related to Command Rules

Report	Description
Command Rule Audit Report	Lists audit records generated by command rule processing operations
Command Rule Configuration Issues Report	Tracks rule violations, in addition to other configuration issues the command rule may have
Object Privilege Reports	Lists object privileges that the command rule affects
Sensitive Objects Reports	Lists objects that the command rule affects
Rule Set Configuration Issues Report	Lists rules sets that have no rules defined or enabled, which may affect the command rules that use them

You can use the `DBA_DV_COMMAND_RULE` data dictionary view to find the SQL statements that are protected by command rules. See DBA_DV_COMMAND_RULE View for more information.

8
Configuring Factors

Factors enable you to base Database Vault restrictions on attributes such as a client IP address or a domain.

- What Are Factors?
  A factor is a named variable or attribute, such as a database IP address, that Oracle Database Vault can recognize.
- Default Factors
  Oracle Database Vault provides a set of default factors.
- Creating a Factor
  In general, to create a factor, you first create the factor itself, and then you edit the factor to include its identity.
- Adding an Identity to a Factor
  After you create a new factor, you optionally can add an identity to it.
- Deleting a Factor
  Before you delete a factor, you must remove references to the factor.
- How Factors Work
  Oracle Database Vault processes factors when a session is established.
- Tutorial: Preventing Ad Hoc Tool Access to the Database
  This tutorial demonstrates how to use factors to prevent ad hoc tools (such as SQL*Plus) from accessing the database.
- Tutorial: Restricting User Activities Based on Session Data
  This tutorial shows how to restrict user activities based on their session data, such as the domain the user is using.
- Guidelines for Designing Factors
  Oracle provides guidelines for designing factors.
- How Factors Affect Performance
  The complexity of factors affects the performance of your Oracle database instance.
- Factor Related Reports and Data Dictionary Views
  Oracle Database Vault provides reports and data dictionary views that display information about factors and their identities.

What Are Factors?

A factor is a named variable or attribute, such as a database IP address, that Oracle Database Vault can recognize.

You can use factors for activities such as authorizing database accounts to connect to the database or creating filtering logic to restrict the visibility and manageability of data.

Oracle Database Vault provides a selection of factors that lets you set controls on such components as the domain for your site, IP addresses, databases, and so on. You also can create custom factors, using your own PL/SQL retrieval methods.

Note the following:

- You can use factors in combination with rules in rule sets. The `DVF` factor functions are factor-specific functions that you can use in rule expressions.
- Factors have values (identities) and are further categorized by their factor types. See "Factor Type" in Completing the General Page for Factor Creation for information about factor types.
- You also can integrate factors with Oracle Label Security labels.
- You can run reports on the factors that you create in Oracle Database Vault. See for more information.
- In a multitenant environment, you only can create factors in a PDB, not in the CDB root or the application root.

This chapter explains how to configure factors by using Oracle Database Vault Administrator. Alternatively, you can use the Oracle Database Vault factor APIs to configure factors.

Related Topics

- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the `DVF` schema functions when you use the `DBMS_MACADM` PL/SQL package to manage the various factors.
- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

Default Factors

Oracle Database Vault provides a set of default factors.

For each of these factors, there is an associated function that retrieves the value of the factor. See Oracle Database Vault DVF PL/SQL Factor Functions for a listing of these functions.

You can create custom factors by using your own PL/SQL retrieval methods. A useful PL/SQL function you can use (which is used for many of the default factors) is the `SYS_CONTEXT` SQL function, which retrieves data about the user session. For example, you can use the `CLIENT_PROGRAM_NAME` attribute of `SYS_CONTEXT` to find the name of the program used for the database session. After you create the custom factor, you can query its values similar to the functions used to query the default factors.

See *Oracle Database SQL Language Reference* for more information about the `SYS_CONTEXT` function.

You can use the default factors in your own security configurations. If you do not need them, you can remove them. (That is, they are not needed for internal use by Oracle Database Vault.)

The default factors are as follows:

- **Authentication_Method:** Is the method of authentication. In the list that follows, the type of user is followed by the method returned:

  - Password-authenticated enterprise user, local database user, user with the SYSDBA or SYSOPER administrative privilege using the password file; proxy with user name using password: PASSWORD
  - Kerberos-authenticated enterprise user or external user (with no administrative privileges): KERBEROS
  - Kerberos-authenticated enterprise user (with administrative privileges): KERBEROS_GLOBAL
  - Kerberos-authenticated external user (with administrative privileges): KERBEROS_EXTERNAL
  - SSL-authenticated enterprise or external user (with no administrative privileges): SSL
  - SSL-authenticated enterprise user (with administrative privileges): SSL_GLOBAL
  - SSL-authenticated external user (with administrative privileges): SSL_EXTERNAL
  - Radius-authenticated external user: RADIUS
  - OS-authenticated external user, or user with the SYSDBA or SYSOPER administrative privilege: OS
  - Proxy with certificate, DN, or username without using password: NONE
  - Background process (job queue slave process): JOB
  - Parallel Query Slave process: PQ_SLAVE

  For non-administrative connections, you can use the Identification_Type factor to distinguish between external and enterprise users when the authentication method is PASSWORD, KERBEROS, or SSL. For administrative connections, the Authentication_Method factor is sufficient for the PASSWORD, SSL_EXTERNAL, and SSL_GLOBAL authentication methods.

- **Client Identifier:** An identifier that is set by the application through the DBMS_SESSION.SET_IDENTIFIER procedure, the Oracle Call Interface (OCI) attribute OCI_ATTR_CLIENT_IDENTIFIER, or Oracle Dynamic Monitoring Service (DMS). Various Oracle Database components use this attribute to identify lightweight application users who authenticate as the same database user.
- **Client_IP:** Is the IP address of the machine from which the client is connected.
- **Database_Domain:** Is the domain of the database as specified in the DB_DOMAIN initialization parameter.
- **Database_Hostname:** Is the host name of the computer on which the instance is running.
- **Database_Instance:** Is the instance identification number of the current instance.
- **Database_IP:** Is the IP address of the computer on which the instance is running.
- **Database_Name:** Is the name of the database as specified in the DB_NAME initialization parameter.
- **DBlink_Info:** The source of a database link session. The string has this form:

  SOURCE_GLOBAL_NAME=*dblink_src_global_name*, DBLINK_NAME=*dblink_name*,SOURCE_AUDIT_SESSIONID=*dblink_src_audit_sessionid*

  In this specification:

  - *dblink_src_global_name* is the unique global name of the source database
  - *dblink_name* is the name of the database link on the source database
  - *dblink_src_audit_sessionid* source database that initiated source database that initiated the connection to the remote database using *dblink_name*

- **Domain:** Is a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level. You can identify a domain using factors such as host name, IP address, and database instance names of the Database Vault nodes in a secure access path to the database. Each domain can be uniquely determined using a combination of the factor identifiers that identify the domain. You can use these identifying factors and possibly additional factors to define the Maximum Security Label within the domain. This restricts data access and commands, depending on the physical factors about the Database Vault session. Example domains of interest may be Corporate Sensitive, Internal Public, Partners, and Customers.
- **Enterprise_Identity:** Is the enterprise-wide identity for the user:
  - For enterprise users: the Oracle Internet Directory-distinguished name (DN).
  - For external users: the external identity (Kerberos principal name, Radius and DCE schema names, operating system user name, certificate DN).
  - For local users and `SYSDBA` and `SYSOPER` logins: NULL.

  The value of the attribute differs by proxy method:

  - For a proxy with DN: the Oracle Internet Directory DN of the client.
  - For a proxy with certificate: the certificate DN of the client for external users; the Oracle Internet Directory DN for global users.
  - For a proxy with user names: the Oracle Internet Directory DN if the client is an enterprise user; NULL if the client is a local database user.
- **Identification_Type:** Is the way the user schema was created in the database. Specifically, it reflects the `IDENTIFIED` clause in the `CREATE USER` and `ALTER USER` syntax. In the list that follows, the syntax used during schema creation is followed by the identification type returned:
  - `IDENTIFIED BY` *password*: `LOCAL`
  - `IDENTIFIED EXTERNALLY`: `EXTERNAL`
  - `IDENTIFIED GLOBALLY`: `GLOBAL SHARED`
  - `IDENTIFIED GLOBALLY AS DN`: `GLOBAL PRIVATE`
- **Lang:** Is the ISO abbreviation for the language name, a shorter form than the existing `LANGUAGE` parameter.
- **Language:** Is the language and territory your session currently uses, along with the database character set, in the following form:

  language_territory.characterset

  For example:

  ```
  AMERICAN_AMERICA.WE8MSWIN1252
  ```

  Refer to *Oracle Database Globalization Support Guide* for more information about languages, territories, and character sets.

- **Machine:** Is the host name for the database client that established the current session. If you must find out whether the computer was used for a client or server session, then you can compare this setting with the Database_Hostname factor to make the determination.
- **Module:** The application name (module) that is set through the `DBMS_APPLICATION_INFO` PL/SQL package or OCI.
- **Network_Protocol:** Is the network protocol being used for communication, as specified in the `PROTOCOL=protocol` portion of the connect string.
- **Proxy_Enterprise_Identity:** Is the Oracle Internet Directory DN when the proxy user is an enterprise user.
- **Proxy_User:** Is the name of the database user who opened the current session on behalf of `SESSION_USER`.
- **Session_User:** Is the database user name by which the current user is authenticated. This value remains the same throughout the session.

Creating a Factor

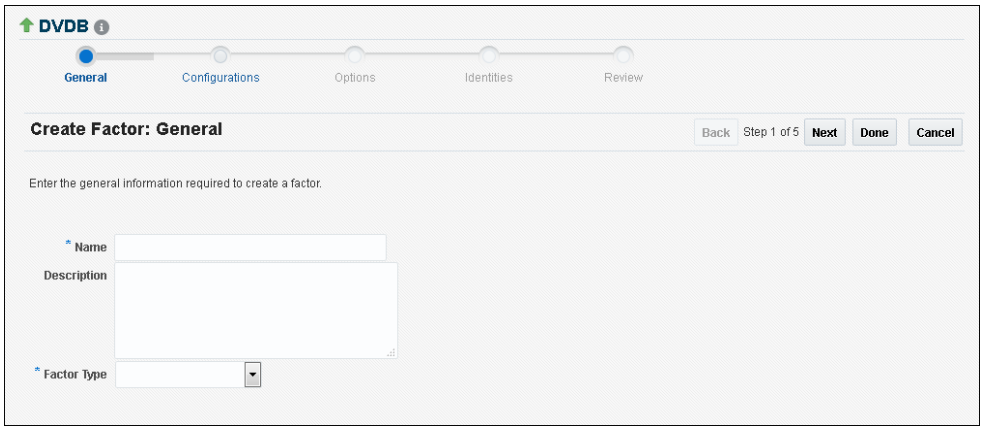
In general, to create a factor, you first create the factor itself, and then you edit the factor to include its identity.

- Accessing the Create Factors Page
  The Create Factors page enables you to create the factor, starting with a general definition of the factor that you want to create.
- Completing the General Page for Factor Creation
  In the General page, you must enter general identifying information for the factor, such as its name.
- Configurations Page for Factor Creation
  The Configurations page defines settings such as the factor's identification and the evaluation method.
- Options Page of Factor Creation
  The Options page assigns a rule set to a factor, sets error options, and for non-unified auditing, sets audit options.

Accessing the Create Factors Page

The Create Factors page enables you to create the factor, starting with a general definition of the factor that you want to create.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Factors**.
3. In the Factors page, click **Create** to display the Create Factor page.



4. Starting with the General page, enter the following information, clicking **Next** to go to each subsequent page, and then clicking **Done** and **Finish** when the factor definition is complete.
   - Completing the General Page for Factor Creation
   - Configurations Page for Factor Creation
   - Options Page of Factor Creation
   - Creating and Configuring a Factor Identity

Completing the General Page for Factor Creation

In the General page, you must enter general identifying information for the factor, such as its name.

- In the General page, enter the following information:
  - **Name:** Enter a name up to 28 characters in mixed-case, without spaces. Oracle Database Vault creates a valid Oracle identifier for the factor function to be created in the `DVF` schema based on the name of the factor chosen. For example, if you create a factor named `GetNetworkIP`, Oracle Database Vault creates the `DVF.F$GETNETWORKIP` function. This attribute is mandatory.

    Oracle suggests that you start the name with a noun and complete the name with a brief description of the derived value.

    Oracle Database Vault DVF PL/SQL Factor Functions describes the `DVF` factor functions.
  - **Description:** Enter a text description of the factor. It can have up to 1024 characters in mixed-case. This attribute is optional.
  - **Factor Type:** From the list, select the type or category of the factor. This attribute is mandatory.

    Factor types have a name and description and are used only to help classify factors. A factor type is the category name used to classify the factor. The default physical factor types include authentication method, host name, host IP address, instance identifiers, database account information, and others. You can create user-defined factor types, such as application name, certificate information, and so on in addition to the installed factor types, such as time and authentication method.

You can find the factors that are associated with a particular factor type by querying the DBA_DV_FACTOR data dictionary view. For example:

```
SELECT NAME FROM DBA_DV_FACTOR
WHERE FACTOR_TYPE_NAME='Authentication Method';
```

The output is:

```
NAME
------------------------------
Network_Protocol
Authentication_Method
Identification_Type
```

Configurations Page for Factor Creation

The Configurations page defines settings such as the factor's identification and the evaluation method.

- Setting the Factor Identification Information
  Under Factor Identification, you must select how to resolve the identity of a factor. This attribute is mandatory.
- How Factor Identities Work
  A factor identity is the actual value of a factor (for example, the IP address for a factor that uses the IP_Address type).
- Setting the Evaluation Information for a Factor
  Under Evaluation, you must select how you want the factor to be evaluated and assigned an identity.
- Setting the Oracle Label Security Labeling Information for a Factor
  Under Factor Labeling, you must select how you want the factor identity to retrieve an Oracle Label Security (OLS) label.
- Setting the Retrieval Method for a Factor
  Under Retrieval Method, you must enter a PL/SQL expression that retrieves the identity of a factor or a constant.
- How Retrieval Methods Work
  The Retrieval Method identifies factors where the factor identification is by method or constant.
- Setting the Validation Method for a Factor
  A validation method uses a PL/SQL expression to return a Boolean value to validate the identity of a factor.

Setting the Factor Identification Information

Under Factor Identification, you must select how to resolve the identity of a factor. This attribute is mandatory.

- In the Configurations page, under Factor Identification, enter the following information:
  - **By Constant:** Resolves the factor identity by retrieving the constant value found in the **Retrieval Method** field.
  - **By Method:** Sets the factor identity by executing the PL/SQL expression specified in the **Retrieval Method** field.

For example, suppose the expression retrieves the system date:

```
to_char(sysdate,'yyyy-mm-dd')
```

On December 15, 2015, the **By Method** option would return the following value:

```
2015-12-15
```

– **By Factors:** Determines the factor identity by mapping the identities of the child factor to its parent factor. A parent factor is a factor whose values are resolved based on a second factor, called a child factor. To establish their relationship, you map their identities. (You do not need to specify a **Retrieval Method** expression for this option.)

See Using Identity Mapping to Configure an Identity to Use Other Factors for more information about mapping identities.

How Factor Identities Work

A factor identity is the actual value of a factor (for example, the IP address for a factor that uses the IP_Address type).

A factor can have several identities depending on its retrieval method or its identity mapping logic. For example, a factor such as Database_Hostname could have multiple identities in an Oracle Real Application Clusters environment; a factor such as Client_IP can have multiple identities in any RDBMS environment. The retrieval method for these types of factors may return different values because the retrieval method is based on the database session. Several reports allow you to track the factor identity configuration.

You can configure the assignment of a factor in the following ways:

- Assign the factor at the time a database session is established.
- Configure individual requests to retrieve the identity of the factor.

With the Oracle Label Security integration, you can label identities with an Oracle Label Security label. You can also assign an identity *trust levels*, which are numbers that indicate the magnitude of trust relative to other identities for the same factor. In general, the higher the trust level number is set, the greater the trust. Negative trust levels are not trusted.

Within a database session, a factor assigned identity is available to Oracle Database Vault and any application with a publicly accessible PL/SQL function that exists in the `DVF` schema (which contains functions that retrieve factor values) as follows:

`dvf.f$`*`factor_name`*

This allows the identifier for a factor to be accessed globally from within the Oracle database (using PL/SQL, SQL, Oracle Virtual Private Database, triggers, and so on). For example, in SQL*Plus:

```
CONNECT leo_dvowner
Enter password: password

SELECT DVF.F$DATABASE_IP FROM DUAL;
```

Output similar to the following appears:

```
SELECT DVF.F$DATABASE_IP FROM DUAL;

F$DATABASE_IP
------------------------------------------------------------
192.0.2.1
```

You can also use the GET_FACTOR function to find the identity of a factor that is made available for public access. For example:

```
SELECT GET_FACTOR('DATABASE_IP') FROM DUAL;
```

The following output appears:

```
GET_FACTOR('DATABASE_IP')
------------------------------------------------------------
192.0.2.1
```

Related Topics

- Adding an Identity to a Factor
  After you create a new factor, you optionally can add an identity to it.
- Factor Related Reports and Data Dictionary Views
  Oracle Database Vault provides reports and data dictionary views that display information about factors and their identities.

Setting the Evaluation Information for a Factor

Under Evaluation, you must select how you want the factor to be evaluated and assigned an identity.

See How Factors Affect Performance for the performance effect of session factors. This attribute is mandatory.

- In the Configurations page, under Evaluation, enter the following information:
  - **For Session:** Evaluates the factor when a database session is created.
  - **By Access:** Evaluates the factor each time it is accessed (for example, referenced by an application) and when the database session is first created.
  - **On Startup:** Evaluates the factor when the database session starts.

Setting the Oracle Label Security Labeling Information for a Factor

Under Factor Labeling, you must select how you want the factor identity to retrieve an Oracle Label Security (OLS) label.

This setting applies if you plan to use the Oracle Label Security integration. This attribute is mandatory if you want to use an OLS label.

- In the Configurations page, under Factor Labeling, enter the following information:
  - **By Self:** Labels the identities for the factor directly from the labels associated with an Oracle Label Security policy.
  - **By Factors:** If there are multiple child factor labels, then Oracle Database Vault merges the labels by using the Oracle Label Security algorithm that is associated with the applicable Oracle Label Security policy. For each applicable Oracle Label Security policy, a factor identity can have an assigned label.

Related Topics

- Integrating Oracle Database Vault with Oracle Label Security
  You can integrate Oracle Database Vault with Oracle Label Security, and check the integration with reports and data dictionary views.

Setting the Retrieval Method for a Factor

Under Retrieval Method, you must enter a PL/SQL expression that retrieves the identity of a factor or a constant.

- In the Configurations page, under Retrieval Method, enter a PL/SQL retrieval method. It can use up to 255 characters in mixed-case.

The following retrieval method sets a value of the DB_NAME factor by retrieving the database name (DB_NAME) from the `USERENV` namespace in a user's session.

```
UPPER(SYS_CONTEXT('USERENV','DB_NAME'))
```

How Retrieval Methods Work

The Retrieval Method identifies factors where the factor identification is by method or constant.

If the factor identification is by factors, Oracle Database Vault identifies it by its identity mappings. You can create your own PL/SQL retrieval methods, or use the functions supplied with Oracle Database Vault. See the following sections for factor-specific and general utility functions that you can use to build the retrieval method:

- Oracle Database Vault DVF PL/SQL Factor Functions
- DBMS_MACADM Factor Procedures and Functions
- Oracle Database Vault Utility APIs

See also the default factors provided with Oracle Database Vault for examples of retrieval methods. Default Factors describes these factors.

The **Retrieval Method** field is mandatory if you have selected the following settings under Factor Identification:

- **By Method:** Enter a method in the Retrieval Method field.
- **By Constant:** Enter a constant in the Retrieval Method field.

The value returned as the factor identity must be a `VARCHAR2` string or otherwise convertible to one.

You can include any package function or standalone function in the expression. Ensure that the expression is a fully qualified function, such as *`schema.function_name`*. Do not include complete SQL statements. If you are using application packages or functions, you must provide `DVSYS` with the `EXECUTE` privilege on the object.

Write the function signature using the following format:

```
FUNCTION GET_FACTOR RETURN VARCHAR2
```

Setting the Validation Method for a Factor

A validation method uses a PL/SQL expression to return a Boolean value to validate the identity of a factor.

Under Validation Method, you must enter a PL/SQL expression that returns a Boolean value (`TRUE` or `FALSE`) to validate the identity of a factor being retrieved (with the `GET_FACTOR` function) or the value to be assigned to a factor (with the `SET_FACTOR` function).

If the method is evaluated to false for the value being retrieved or to be assigned, then the factor identity is set to null. This optional feature provides an additional level of assurance that the factor is properly retrieved and set. This field can have up to 255 characters in mixed-case.

You can include any package function or standalone function in the expression. Ensure that the expression is a fully qualified function, such as *`schema.function_name`*. Do not include complete SQL statements. If you are using application packages or functions, then you must provide `DVSYS` with the `EXECUTE` privilege on the object.

- In the Configurations page, under Validation method, create a function that uses any of the following formats:
  - `FUNCTION IS_VALID RETURN BOOLEAN`

    In this form, you can use the `DVF.F$`*`factor_name`* function inside the function logic. This is more appropriate for factors that are evaluated by session.
  - `FUNCTION IS_VALID(`*`p_factor_value`* `VARCHAR2) RETURN BOOLEAN`

    In this form, the factor value is passed to the validation function directly. This is more appropriate for factors that are evaluated by access. It is also valid for factors evaluated by session.

Related Topics

- DBMS_MACADM Factor Procedures and Functions
  The `DBMS_MACADM` PL/SQL package provides procedures and functions to configure factors.
- Oracle Database Vault Run-Time PL/SQL Procedures and Functions
  Oracle Database Vault provides procedural interfaces to administer Database Vault security options and manage Database Vault security enforcements.
- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the `DVF` schema functions when you use the `DBMS_MACADM` PL/SQL package to manage the various factors.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

Options Page of Factor Creation

The Options page assigns a rule set to a factor, sets error options, and for non-unified auditing, sets audit options.

- Assigning a Rule Set to a Factor
  Under Assignment Rule Set, you select a rule set if you want to use a rule set to control setting a factor identity.
- Setting Error Options for a Factor
  Under Error Options, you set the processing that must occur when a factor identity cannot be resolved.
- Setting Audit Options for a Factor
  Under Audit Options, you can generate an audit trail if you are not using a unified audit environment.
- How Factor Auditing Works
  Whether you have unified auditing enabled affects how auditing is handled for factors.

Assigning a Rule Set to a Factor

Under Assignment Rule Set, you select a rule set if you want to use a rule set to control setting a factor identity.

For example, you can use a rule set to determine when a database session originates from a known application server or program.

- In the Options page, from the **Assignment Rule Set** menu, select a rule set from the list.

This attribute is particularly useful for situations where database applications, such as a Web application using a JDBC connection pool, must dynamically set a factor identity for the current database session. For example, a Web application may want to assign the geographic location for a database account logging in to the Web application. To do so, the Web application can use the JDBC Callable Statement, or Oracle Data Provider for .NET (ODP.NET) to execute the PL/SQL function `SET_FACTOR`, for example:

```
BEGIN
 SET_FACTOR('GEO_STATE','VIRGINIA');
END;
```

Then you can create an assignment rule for the GEO_STATE factor to allow or disallow the setting of the GEO_STATE factor based on other factors or rule expressions.

Related Topics

- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.
- How Factors Are Set
  You can assign a factor identity at any time during a database session, but only if the factor assignment rule set evaluates to true.

Setting Error Options for a Factor

Under Error Options, you set the processing that must occur when a factor identity cannot be resolved.

- In the Options page, under Error Options, select from the following values:

  - **Show Error Message:** Displays an error message to the database session.
  - **Do Not Show Error Message:** Does not display the error message.

    An advantage of selecting **Do Not Show Error Message** and then enabling auditing is that you can track the activities of a potential intruder. The audit report reveals the activities of the intruder, yet the intruder is unaware that you are doing this because he or she does not see any error messages.

After you have created a new factor, you are ready to configure its identity. To do so, edit the factor and then add its identity.

Setting Audit Options for a Factor

Under Audit Options, you can generate an audit trail if you are not using a unified audit environment.

- In the Options page, under Audit Options, select from the following values:
  - **Never:** Does not audit.
  - **Always:** Always creates an audit record when a factor is evaluated. You can select from the conditions, described next.
  - **Validation False:** Creates an audit record when the validation method (if provided) returns `FALSE`.
  - **Retrieval Error:** Creates an audit record when the identity of a factor cannot be resolved and assigned, due to an error (such as `No data found` or `Too many rows`).
  - **Trust Level NULL:** Creates an audit record when the resolved identity of a factor has an assigned trust level of `NULL`.

    See Creating and Configuring a Factor Identity for more information about trust levels.
  - **Trust Level Less Than Zero:** Creates an audit record when the resolved identity of a factor has an assigned trust level less than zero.
  - **Validation Error:** Creates an audit record when the validation method (if provided) returns an error.

How Factor Auditing Works

Whether you have unified auditing enabled affects how auditing is handled for factors.

In a non-unified auditing environment, Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` table, described in Auditing Oracle Database Vault.

If you have enabled unified auditing, then this setting does not capture audit records. Instead, you can create audit policies to capture this information, as described in *Oracle Database Security Guide*.

You can use the Factor Audit Report to display the generated audit records. (See Factor Related Reports and Data Dictionary Views for more information.) In addition, you can select multiple audit options at a time. Each option is converted to a bit mask and added to determine the aggregate behavior. Note that there is little performance impact in auditing, unless the factor has errors.

Adding an Identity to a Factor

After you create a new factor, you optionally can add an identity to it.

- About Factor Identities
  An identity is the actual value of a factor, such an IP_Address factor identity being 192.0.2.4.
- About Trust Levels
  Trust levels enable you to assign a numeric value to indicate the measure of trust allowed.
- About Label Identities
  You can assign You Oracle Label Security (OLS) labels to factor identities.
- Creating and Configuring a Factor Identity
  You can create and configure a factor identity in Oracle Database Vault Administrator.
- Deleting a Factor Identity
  If you want to delete a factor identity, you can locate references to it by querying the factor-related Oracle Database Vault views.
- Using Identity Mapping to Configure an Identity to Use Other Factors
  You can use identity mapping to use a group of factors to manage identity values.

About Factor Identities

An identity is the actual value of a factor, such an IP_Address factor identity being 192.0.2.4.

A factor identity for a given database session is assigned at run time using the **Factor Identification** and **Retrieval Method** fields described in Creating a Factor. You can further configure the identity for the following reasons:

- To define the known identities for a factor
- To add a trust level to a factor identity
- To add an Oracle Label Security label to a factor identity
- To resolve a factor identity through its child factors, by using identity mapping

Related Topics

- Tutorial: Restricting User Activities Based on Session Data
  This tutorial shows how to restrict user activities based on their session data, such as the domain the user is using.

About Trust Levels

Trust levels enable you to assign a numeric value to indicate the measure of trust allowed.

A trust value of 1 signifies some trust. A higher value indicates a higher level of trust. A negative value or zero indicates distrust. When the factor identity returned from a factor retrieval method is not defined in the identity, Oracle Database Vault automatically assigns the identity a negative trust level.

To determine the trust level of a factor identity at run time, you can use the GET_TRUST_LEVEL and GET_TRUST_LEVEL_FOR_IDENTITY functions in the DVSYS schema.

For example, suppose you have created a factor named Network. You can create the following identities for the Network factor:

- Intranet, with a trust level of 10
- VPN (virtual private network), with a trust level of 5
- Public, with a trust level of 1

You then can create rule expressions (or custom application code) that base policy decisions on the trust level. For example, you can use the GET_TRUST_LEVEL function to find trust levels greater than 5:

```
GET_TRUST_LEVEL('Network') > 5
```

Or, you can use a SELECT statement on the DBA_DV_IDENTITY data dictionary view to find trust levels for the Network factor greater than or equal to 5:

```
SELECT VALUE, TRUST_LEVEL FROM DBA_DV_IDENTITY
   WHERE TRUST_LEVEL >= 5
   AND FACTOR_NAME='Network'
```

Output similar to the following appears:

```
F$NETWORK GET_TRUST_LEVEL('NETWORK')
-----------------------------------
VPN                                5
INTRANET                          10
```

In the preceding example, Network factor identity for VPN is trusted (value equals 5), and the identity for the INTRANET domain is 10, which implies a greater trust.

Related Topics

- Oracle Database Vault Realm APIs
  The DBMS_MACADM PL/SQL package enables you to configure Oracle Database Vault realms.

About Label Identities

You can assign You Oracle Label Security (OLS) labels to factor identities.

In brief, a label acts as an identifier for a database table row to assign privileges to the row. The **Factor Labeling** attribute for a factor determines whether a factor is labeled **By Self** or **By Factors**. If you set the **Factor Labeling** attribute to **By Self**, then you can associate OLS labels with the factor identities. If you set the **Factor Labeling** attribute to **By Factors**, then Oracle Database Vault derives the factor identity labels from the labeling of child factor identities. When there are multiple child factor identities with labels, Oracle Database Vault merges the labels using the OLS algorithm associated with the applicable factor Oracle Label Security policy.

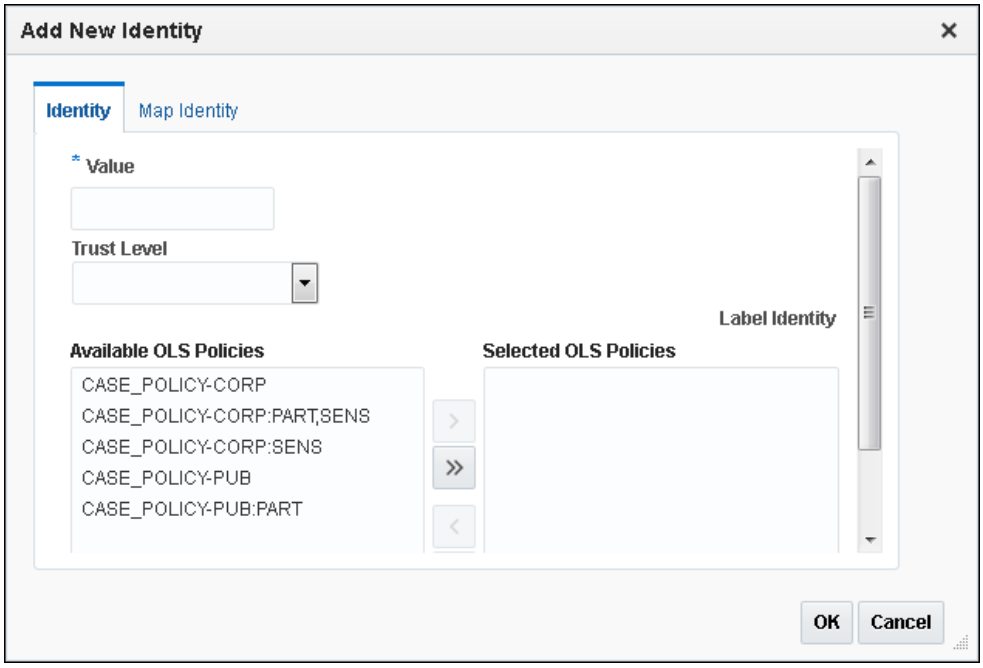
> **See Also:**
>
> *Oracle Label Security Administrator's Guide* for more information about labels

Creating and Configuring a Factor Identity

You can create and configure a factor identity in Oracle Database Vault Administrator.

1. In the Select Identities page of the Create Factor pages, select **Add New Identity.**

   The Add New Identity window appears.



2. In the Identity subpage, enter the following values:
   - **Value:** Enter the value of the identity, up to 1024 characters in mixed-case. This attribute is mandatory.
   - **Trust Level:** Select one of the following trust levels:
     - **Very Trusted:** Assigns a trust level value of 10
     - **Trusted:** Assigns a trust level value of 5
     - **Somewhat Trusted:** Assigns a trust level value of 1
     - **Untrusted:** Assigns a trust level value of -1
     - **Trust Level Not Defined:** Assigns a trust level value of `NULL` (default)

     See About Trust Levels for detailed information about trust levels.
   - **Label Identity:** Optionally, select from the list of available Oracle Label Security policies and then click the **Move** button to move them to the **Selected OLS Policies** list.

The list shows data labels from the Oracle Label Security installation for your site. For more information, refer to *Oracle Label Security Administrator's Guide*.

See About Label Identities for detailed information about label identities.

3. Click **OK** to return to the Create Factors : Identities page..
4. Click **Next** to review the factor settings.
5. Click **Finish**.

Deleting a Factor Identity

If you want to delete a factor identity, you can locate references to it by querying the factor-related Oracle Database Vault views.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Factors**.
3. Select the factor whose identity you want to delete, and then click **Edit**.
4. In the Edit Factor page, click **Next** until you reach the Identities page.
5. Select the factor identity that you want to remove.
6. Click **Remove**.
7. Click **Done**, then click **Finish**.

Related Topics

- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

Using Identity Mapping to Configure an Identity to Use Other Factors

You can use identity mapping to use a group of factors to manage identity values.

- About Identity Mapping
  While you are creating a factory identity, you can map it.
- Mapping an Identity to a Factor
  You can map an identity to a factor by creating a parent-child relationship with two factors.

About Identity Mapping

While you are creating a factory identity, you can map it.

Identity mapping is the process of identifying a factor by using other (child) factors. This is a way to transform combinations of factors into logical identities for a factor or to transform continuous identity values (for example, temperature) or large discrete identity values (for example, IP address ranges) into logical sets. To check configuration issues in the mapping for an identity, you can run the Identity Configuration Issues report.

You can map different identities of a parent factor to different identities of the contributing factor. For example, an INTRANET identity maps to an IP address range of 192.0.2.1 to 192.0.2.24. A REMOTE identity can map to an IP address range that excludes the address range 192.0.2.1 to 192.0.2.24.

Based on identity mapping, you can create a security policy. For example, you can define a reduced set of privileges for an employee connecting over VPN (with REMOTE), as opposed to an employee connecting from within the corporate network (with INTRANET).

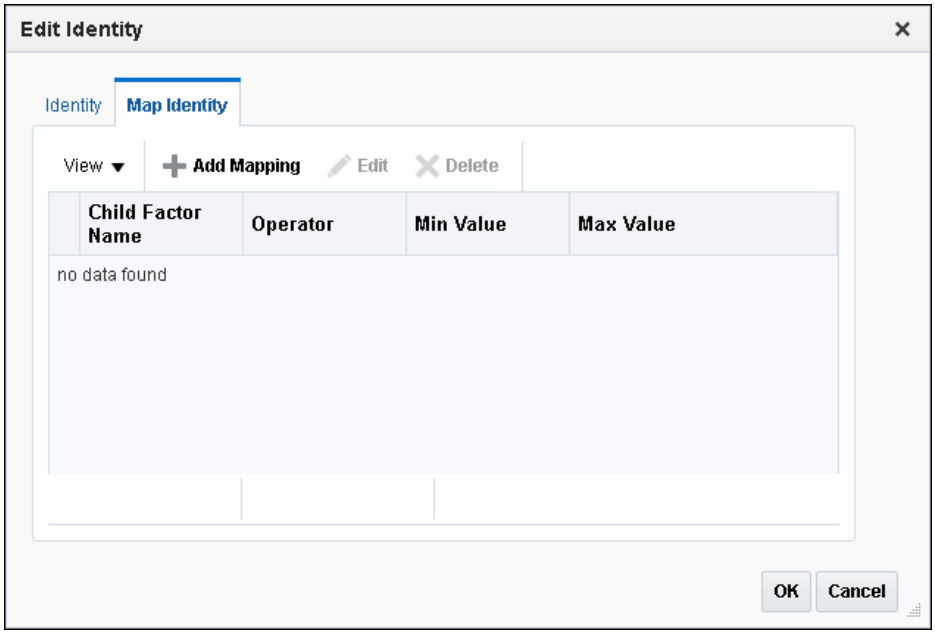
Related Topics

- Tutorial: Restricting User Activities Based on Session Data
  This tutorial shows how to restrict user activities based on their session data, such as the domain the user is using.

Mapping an Identity to a Factor

You can map an identity to a factor by creating a parent-child relationship with two factors.

1. Follow the instructions in Creating a Factor to create a parent factor and set the attribute **Factor Identification** to **By Factors**.
2. In the Identities page, follow the instructions in Creating and Configuring a Factor Identity to create an identity for the parent factor.
3. Map the factor-identity pair of the parent to the factor-identity pairs of its children. Use the following procedure:
   a. In the Identities page, either select an existing identity and click **Edit**, or click **Add New Identity** to create a new identity.
   b. In the Edit Identity window (or the Add New Identity window), ensure that at least the **Value** field is filled out in the Identity subpage.
   c. Click the **Map Identity** tab.



d. Click **Add Mapping**.

e. Enter the following information:

Child Factor Name: From the list, select the child factor name.

Operator: Select the operator from the list.

Min Value: Enter the minimum value.

Max Value: Enter the maximum value.

For example, consider a scenario where the Contributing Factor to the Factor Network is set to Client_IP, the **Operator** is set to `Between`, the **Min Value** is set to 192.0.2.1 and the **Max Value** is set to 192.0.2.24. This means that whenever the client IP address lies in the specified address range of 192.0.2.1 to 192.0.2.24, the parent factor evaluates to a predefined identity (for example, INTRANET).

f. Click **OK** to exit the Add New Identity Mapping window.

g. Click **OK** to exit the Add New Identity and Mapping window.

4. Click **Done**, and then click **Finish**.

Repeat this process to add more contributing factors for a parent factor identity. For example, you can configure the Network factor to resolve to a value ACCOUNTING-SENSITIVE, when the Program factor resolves to "Oracle General Ledger" and the Client_IP is in between 192.0.2.1 and 192.0.2.24. So, if an authorized accounting financial application program, running on a client with IP address 192.0.2.12 accesses the database, then the Network factor is resolved to ACCOUNTING-SENSITIVE. A database session with the ACCOUNTING-SENSITIVE Network value would have more access privileges than one with the INTRANET Network value.

Deleting a Factor

Before you delete a factor, you must remove references to the factor.

You can find the various references to the factor and its identities by querying the factor-related Oracle Database Vault views. See Oracle Database Vault Data Dictionary Views, for more information.

1. Delete any references to the factor, such as rule sets, factor identities, and Oracle Label Security policy associations.

   To do so, edit the factor. You can find and remove the rule set from the Options page and the Oracle Label Security policies associations and identities from the Identities page.

2. In the Oracle Database Vault Administration page, select **Factors**.
3. In the Factors page, select the factor that you want to remove.
4. Click **Delete**.
5. In the Confirmation window, click **Yes**.

How Factors Work

Oracle Database Vault processes factors when a session is established.

- How Factors Are Processed When a Session Is Established
  Oracle Database Vault evaluates the factors based on when a session begins.
- How Factors Are Retrieved
  You can retrieve a factor in a database session at any time by using the `DVF` factor function or the `GET_FACTOR` function.
- How Factors Are Set
  You can assign a factor identity at any time during a database session, but only if the factor assignment rule set evaluates to true.

How Factors Are Processed When a Session Is Established

Oracle Database Vault evaluates the factors based on when a session begins.

When a database session is established, the following actions occur:

1. At the start of each database session, Oracle Database Vault begins to evaluate all default and user-created factors in the database instance.

   This evaluation occurs after the normal database authentication of the session and the initialization of the Oracle Label Security session information, if applicable.
2. In the factor evaluation stage, the factor initialization process executes the retrieval method for all factors that are identified by methods or constants, to resolve the factor identity for the session.

   The factor error options setting has no effect on the factor initialization process.
3. If a factor has a validation method defined, Oracle Database Vault validates the identity (value) of the factor by executing this validation method. If the validation method fails or returns false, the identity of the factor is undefined (`NULL`).
4. If a factor has any identities defined for it, Oracle Database Vault resolves the trust level of the factor based on the identities defined. If an identity of the factor is defined in this list of defined identities, then Oracle Database Vault assigns the trust level as configured; otherwise it sets it to -1. If there are no identities defined for the factor, the trust level is undefined (`NULL`).
5. Depending on the outcome of this factor evaluation, factor validation, and trust level resolution, Database Vault audits the details of the evaluation as dictated by the factor audit configuration.
6. When the evaluation of all factors that are identified by method or constant completes, Oracle Database Vault resolves the factors that are identified by other factors by using the identity maps that are defined for the factor configured identities.

   The evaluation order of the factor-configured identities is by ASCII sort on the identity values: Oracle Database Vault uses the first alphabetically sorted identity mapping that it evaluates. For example, suppose factor TEST has identities X and Y. Furthermore, identities X and Y have identity maps that are dependent on identities for factors A, B, and C. The following mapping occurs:

   - X is mapped when A=1 and B=1
   - Y is mapped when A=1, B=1, and C=2

   In this case, the first one evaluated is X. Y is not evaluated, but what if its C mapping meets the criteria that is needed for the TEST factor's success? You would need to reverse the mapping, that is, map Y before X so that A, B, and C

can be evaluated first. To reverse the mapping, rename Y to V (or some alphabetic value that sorts before X) so that it can be correctly resolved.

This algorithm works if the ASCII sort ordering is correct and the identities map the same number factors at some level.

7. When the factor initialization completes, the Oracle Database Vault integration with Oracle Label Security occurs.

After this process completes, Oracle Database Vault checks to see if a command rule is associated with the `CONNECT` event. If a rule set associated with the `CONNECT` event, then Oracle Database Vault evaluates the rule set. If the rule set evaluates to false or results in an error, then the session is terminated. Oracle Database Vault executes any auditing or call handlers associated with the rule set before the session is terminated.

> **Note:**
>
> Be careful about associating command rules with the `CONNECT` event, because you can inadvertently lock out other users from of the database. In general, if you create a command rule for `CONNECT`, set its evaluation option of the associated rule set to `Any True`.
>
> If you do inadvertently lock out users, then you should temporarily disable Oracle Database Vault, disable the CONNECT command rule, reenable Oracle Database Vault, and then fix the factor code that is causing the problem. If the Test Fails provides an example of how to accomplish this.

How Factors Are Retrieved

You can retrieve a factor in a database session at any time by using the `DVF` factor function or the `GET_FACTOR` function.

To find a listing of available factors, query the `DBA_DV_FACTOR` data dictionary view, described in DBA_DV_FACTOR View.

Example 8-1 shows an example of using the `GET_FACTOR` function.

Example 8-1 Using GET_FACTOR to Retrieve a Factor

```
SELECT GET_FACTOR('client_ip') FROM DUAL;
```

You can use the factor values retrieved from the `DVF` factor function or the `GET_FACTOR` in the following ways:

- Oracle Database Vault rule expressions
- Custom application code that is available to all database sessions in an Oracle Database Vault environment

Oracle Database Vault DVF PL/SQL Factor Functions describes `DVF` factor functions in detail.

If you had set the factor evaluation to **By Session**, then Oracle Database Vault retrieves the value from the session context established, as described under How Factors Are Processed When a Session Is Established.

If you had set the factor evaluation to **By Access**, then Oracle Database Vault performs Step 2 through Step 5 (or Step 6), as described under How Factors Are Processed When a Session Is Established, whenever the factor is retrieved.

If you had defined error options for the factor and if an error occurs, then Oracle Database Vault displays the error message.

How Factors Are Set

You can assign a factor identity at any time during a database session, but only if the factor assignment rule set evaluates to true.

You can do this in the application code by using the `SET_FACTOR` function. In Java code, you can use the JDBC class `java.sql.CallableStatement` to set this value. For example:

```
java.sql.Connection connection ;
...
java.sql.CallableStatement statement =
   connection.prepareCall("{call SET_FACTOR('FACTOR_X', ?)}");
statement.setString(1, "MyValue");
boolean result = statement.execute();
...
```

Applications that can execute Oracle PL/SQL functions can use this procedure (for example, applications written using Oracle Data Provider for .NET (ODP.NET)).

This concept is similar to the standard Oracle `DBMS_SESSION.SET_IDENTIFIER` procedure with an added feature that a rule set controls when a factor value can be set. If the rule set evaluates to true, Steps 2 through 5 under How Factors Are Processed When a Session Is Established occur.

If you have not associated a assignment rule set for the factor or if the rule set returns false (or returns errors), then Oracle Database Vault sends an error message if you attempt to set the factor using the `SET_FACTOR` function.

Tutorial: Preventing Ad Hoc Tool Access to the Database

This tutorial demonstrates how to use factors to prevent ad hoc tools (such as SQL*Plus) from accessing the database.

- About This Tutorial
  Many database applications contain features to explicitly control the actions of a user.
- Step 1: Enable the HR and OE User Accounts
  You must use the `HR` and `OE` accounts later on when you test the Oracle Database Vault components for this tutorial.
- Step 2: Create the Factor
  After you have ensured that the `HR` and `OE` accounts are active, you can create a factor.
- Step 3: Create the Rule Set and Rules
  After you have created the factor, you can create a rule set and rules to work with the factor.

- Step 4: Create the CONNECT Command Rule
  The CONNECT command rule controls the `CONNECT` SQL statement.
- Step 5: Test the Ad Hoc Tool Access Restriction
  You do not need to restart your SQL*Plus session for the Oracle Database Vault changes to take effect.
- Step 6: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

About This Tutorial

Many database applications contain features to explicitly control the actions of a user.

However, an ad hoc query tool, such as SQL*Plus, may not have these controls. As a result, a user could use an ad hoc tool to perform actions in the database that he or she would normally be prevented from performing in a database application. You can use a combination of Oracle Database Vault factors, rule sets, and command rules to prevent unauthorized access to the database by ad hoc query tools.

In the following tutorial, you prevent users HR and OE from using SQL*Plus. To accomplish this, you must create a factor to find the applications on your system and a rule and rule set to limit SQL*Plus to these four users. Then you create a command rule for the `CONNECT` SQL statement, which is associated with the rule set. This factor, Client_Prog_Name, uses the `CLIENT_PROGRAM_NAME` attribute of the `SYS_CONTEXT` SQL function `USERENV` namespace to find the names of the applications that are used to access the current instance of Oracle Database. The `SYS_CONTEXT` SQL function provides many useful methods for finding the state of a user session. `SYS_CONTEXT` is a valuable tool for creating custom factors.

> **See Also:**
>
> *Oracle Database SQL Language Reference* for more information about the `SYS_CONTEXT` function.

Step 1: Enable the HR and OE User Accounts

You must use the `HR` and `OE` accounts later on when you test the Oracle Database Vault components for this tutorial.

1. Log into the database instance as a user who has been granted the `DV_ACCTMGR` role.

   For example:

   ```
   sqlplus bea_dvacctmgr
   Enter password: password
   ```

   In a multitenant environment, you must connect to the appropriate pluggable database (PDB).

   For example:

```
sqlplus bea_dvacctmgr@hrpdb
Enter password: password
```

To find the available PDBs, run the `show pdbs` command. To check the current PDB, run the `show con_name` command.

2. Check the status of the HR account.

```
SELECT USERNAME, ACCOUNT_STATUS FROM DBA_USERS WHERE USERNAME = 'HR';
```

3. If the HR account is expired and locked, then enter the following statement to make it active:

```
ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password;
```

Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

4. Repeat these steps for the OE account.

Step 2: Create the Factor

After you have ensured that the HR and OE accounts are active, you can create a factor.

1. Connect as a user who has been granted the DV_OWNER or DV_ADMIN role.

For example:

```
CONNECT leo_dvowner --Or, CONNECT leo_dvowner@hrpdb
Enter password: password
```

2. Create the factor.

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'Client_Prog_Name',
  factor_type_name  => 'Application',
  description       => 'Stores client program name that connects to database',
  rule_set_name     => NULL,
  validate_expr     => NULL,
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''CLIENT_PROGRAM_NAME''))',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_METHOD,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_SESSION,
  audit_options     => DBMS_MACUTL.G_AUDIT_ON_GET_ERROR,
  fail_options      => DBMS_MACUTL.G_FAIL_SILENTLY);
END;
/
```

In this specification:

- factor_type_name specifies that this is an application-based factor.
- get_expr defines the expression for the factor. This expression calls the SYS_CONTEXT function, using the USERENV namespace and CLIENT_PROGRAM_NAME attribute, to find the programs that are logged into the Oracle database.
- identify_by identifies the factor by method.
- labeled_by labels the identities for the factor directly from the labels associated with an Oracle Label Security policy (default).

- `eval_options` evaluates the factor when the database session is created.
- `audit_options` audits if `get_expr` returns an error.
- `fail_silently` does not show any error messages for the factor.

Step 3: Create the Rule Set and Rules

After you have created the factor, you can create a rule set and rules to work with the factor.

1. Create the Limit SQL*Plus Access rule set as follows:

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Limit SQL*Plus Access',
  description      => 'Limits access to SQL*Plus for Apps Schemas',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'SQL*Plus access not allowed for Apps Schemas',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
  handler          => NULL,
  is_static        => FALSE);
END;
/
```

   In this specification:

   - `fail_options` enables an error message, set by `fail_message`, and error code, set by `fail_code`, to appear if there are errors.
   - `is_static` evaluates the rule set once during the user session. After that, the value is re-used.

2. Find the exact settings for the computer on which you want to apply the policy, based on what the `CLIENT_PROGRAM_NAME` attribute will return.

```
SELECT SYS_CONTEXT('USERENV', 'CLIENT_PROGRAM_NAME') FROM DUAL;
```

   The output should be similar to the following:

```
SYS_CONTEXT('USERENV','CLIENT_PROGRAM_NAME')
----------------------------------------------------------------
sqlplus@nemosity (TNS V1-V3)
```

   For this tutorial, the name of the computer is `nemosity`. The `(TN V1-V3)` output refers to the version of the TNS connector.

3. Create the following rules.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Prevent Apps Schemas Access to SQL*Plus',
  rule_expr  =>'UPPER (DVF.F$CLIENT_PROG_NAME) != ''SQLPLUS@NEMOSITY (TNS V1-
V3)'' AND DVF.F$SESSION_USER IN (''HR'', ''OE'')');
END;
/
BEGIN
 DBMS_MACADM.CREATE_RULE(
```

```
  rule_name  => 'Allow Non-Apps Schemas Access to SQL*Plus',
  rule_expr  =>'DVF.F$SESSION_USER NOT IN (''HR'', ''OE'')');
END;
/
```

The rules translate to the following: "Prevent users `HR` and `OE` from logging into SQL*Plus, but allow other users access."

4. Add the rules to the Limit SQL*Plus Access rule set.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit SQL*Plus Access',
  rule_name     => 'Prevent Apps Schemas Access to SQL*Plus',
  rule_order    => 1);
END;
/
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit SQL*Plus Access',
  rule_name     => 'Allow Non-Apps Schemas Access to SQL*Plus',
  rule_order    => 1);
END;
/
```

The `rule_order` setting is required to enable the procedure to work.

Step 4: Create the CONNECT Command Rule

The CONNECT command rule controls the `CONNECT` SQL statement.

This command rule also applies to logging into SQL*Plus from the command line or other tools your site may use to access SQL*Plus.

- Create the CONNECT command rule as follows:

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'CONNECT',
  rule_set_name   => 'Limit SQL*Plus Access',
  object_owner    => '%',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

In this specification:

- `rule_set_name` associates the Limit SQL*Plus Access rule set with the CONNECT command rule.
- `object_owner` is set to `%` so that the command rule applies to all users.
- `object_name` is set to `%` so that the command rule applies to all objects.
- `enabled` enables the command rule so that it can be used right away.

Step 5: Test the Ad Hoc Tool Access Restriction

You do not need to restart your SQL*Plus session for the Oracle Database Vault changes to take effect.

1. In SQL*Plus, try to connect as user HR:

   ```
   CONNECT HR --Or, CONNECT HR@hrpdb
   Enter password: password
   ```

   The following output should appear:

   ```
   ERROR:
   ORA-47306: 20461: Limit SQL*Plus Access rule set failed
   ```

   User HR should be prevented from using SQL*Plus.

2. Next, try to connect as user OE:

   ```
   CONNECT OE --Or, CONNECT OE@hrpdb
   Enter password: password
   ```

   The following output should appear:

   ```
   ERROR:
   ORA-47306: 20461: Limit SQL*Plus Access rule set failed
   ```

   User OE also should be prevented from using SQL*Plus.

3. Now try to connect as user SYSTEM:

   ```
   CONNECT SYSTEM --Or, CONNECT SYSTEM@hrpdb
   Enter password: password
   Connected.
   ```

   User SYSTEM should be able to log into the database instance. So should SYS, the Database Vault Owner account, and the Database Vault Account Manager account.

If the Test Fails

If you cannot log into the database instance as SYSTEM (or as any of the other administrative users listed in your rule expression), then you are prevented from using SQL*Plus.

You can remedy the problem as follows:

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

   For example:

   ```
   CONNECT sec_admin_owen --Or, CONNECT sec_admin_owen@hrpdb for a PDB
   Enter password: password
   ```

2. Enter the following statement to drop the CONNECT command rule.

   ```
   EXEC DBMS_MACADM.DELETE_COMMAND_RULE ('CONNECT', '%', '%');
   ```

   Even though you have disabled Oracle Database Vault, you still can use its PL/SQL packages and Database Vault Administrator.

3. Check the policy components for any errors and then correct them. Recreate the CONNECT command rule, and then test it.

Step 6: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. Remove the CONNECT command rule.

```
EXEC DBMS_MACADM.DELETE_COMMAND_RULE ('CONNECT', '%', '%');
```

2. Remove the Client_Prog_Name factor.

```
EXEC DBMS_MACADM.DELETE_FACTOR('Client_Prog_Name');
```

3. Remove the Limit SQL*Plus Access rule set.

```
EXEC DBMS_MACADM.DELETE_RULE_SET('Limit SQL*Plus Access');
```

4. Remove the rules.

```
EXEC DBMS_MACADM.DELETE_RULE('Prevent Apps Schemas Access to SQL*Plus');
EXEC DBMS_MACADM.DELETE_RULE('Allow Non-Apps Schemas Access to SQL*Plus');
```

5. If necessary, as a user who has been granted the `DBV_ACCTMGR` role, lock the `HR` and `OE` accounts.

```
CONNECT bea_dvacctmgr --Or, CONNECT amalcolumn_dbacctmgr@hrpdb
Enter password: password

ALTER USER HR ACCOUNT LOCK;
ALTER USER OE ACCOUNT LOCK;
```

Tutorial: Restricting User Activities Based on Session Data

This tutorial shows how to restrict user activities based on their session data, such as the domain the user is using.

- About This Tutorial
  You can use factor identity mapping to set session-based user restrictions for database activities.
- Step 1: Create an Administrative User
  Before you can use this tutorial, you must create an administrative user.
- Step 2: Add Identities to the Domain Factor
  Next, you must add identities to the Domain factor, which is a default factor.
- Step 3: Map the Domain Factor Identities to the Client_IP Factor
  After you have added identities to the domain factory, you can map them to the Client_IP factor.
- Step 4: Create a Rule Set to Set the Hours and Select the Factor Identity
  You must create a rule set to work with the factor that you modified.
- Step 5: Create a Command Rule That Uses the Rule Set
  You must create a command rule that uses the rule set that you created.
- Step 6: Test the Factor Identity Settings
  Test the settings by resetting the system clock, logging in as the `mwaldron` administrative user, and then trying to create a table.

- Step 7: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

About This Tutorial

You can use factor identity mapping to set session-based user restrictions for database activities.

For example, suppose you wanted to restrict administrative access to a database using the following criteria:

- Ensure that the administrator is accessing the database from the correct IP address.
- Limit the database access to the standard business hours of the administrator.

This type of configuration is useful for restricting different types of administrators: not only local, internal administrators, but offshore and contract administrators as well.

In this tutorial, you modify the Domain factor to include identities for a secure and non-secure network access, which are based on the IP address of the computer the administrator is using. If the administrator tries to perform an action outside the standard working hours or from a different IP address, then Oracle Database Vault prevents him from doing so.

Step 1: Create an Administrative User

Before you can use this tutorial, you must create an administrative user.

1. In SQL*Plus, log in as a user who has been granted the `DV_ACCTMGR` role, and then create the user account `mwaldron`.

   For example:

   ```
   sqlplus bea_dvacctmgr
   Enter password: password

   CREATE USER mwaldron IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

   In a multitenant environment, you must connect to the appropriate pluggable database (PDB).

   For example:

   ```
   sqlplus bea_dvacctmgr@hrpdb
   Enter password: password
   ```

   To find the available PDBs, query the `DBA_PDBS` data dictionary view. To check the current PDB, run the `show con_name` command.

2. Connect as a user who privileges to grant the `CREATE SESSION` privilege and the `DBA` role, and then grant user `mwaldron` these privileges. This user must also be authorized as an owner of the Oracle System Privilege and Role Management realm.

   For example:

```
CONNECT dba_psmith -- Or, CONNECT dba_psmith@hrpdb
Enter password: password

GRANT CREATE SESSION, DBA TO mwaldron;
```

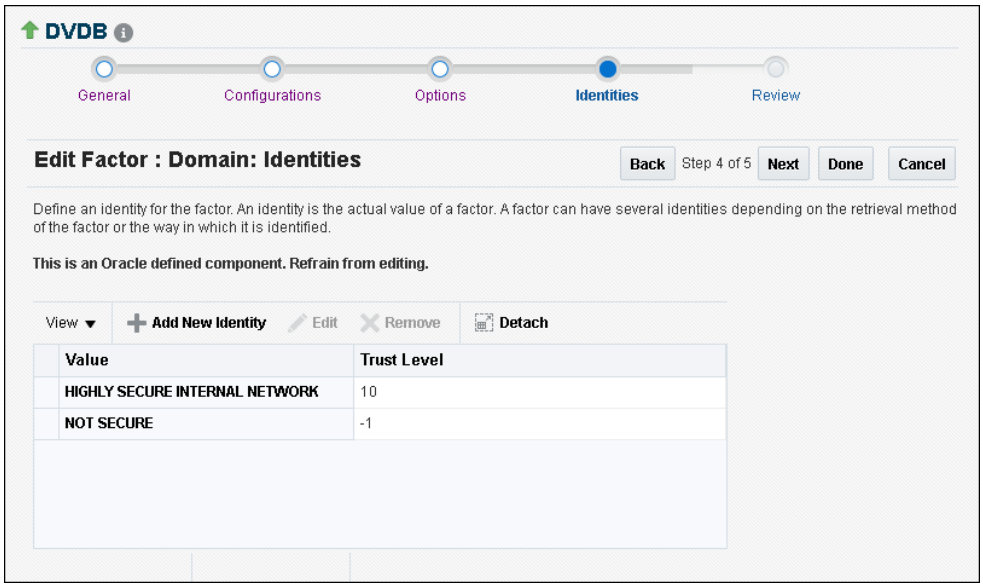
Step 2: Add Identities to the Domain Factor

Next, you must add identities to the Domain factor, which is a default factor.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Factors**.

   The Factors page appears.
3. Select the **Show Oracle defined Factors** check box to display the default factors.
4. Select the Domain factor and then select **Edit**.

   The Domain factor will be the parent factor.
5. Click the **Next** button until you reach the Identities page.
6. Select the **Add New Identity** button.
7. In the Identity tab of the Add New Identity and Mapping page, enter the following information:
   - **Value:** Enter `HIGHLY SECURE INTERNAL NETWORK`
   - **Trust Level:** Select **Very Trusted**
8. Click **OK**.
9. Repeat these steps to create a second identity called `NOT SECURE`, and then set its trust level to **Untrusted**.



Step 3: Map the Domain Factor Identities to the Client_IP Factor

After you have added identities to the domain factory, you can map them to the Client_IP factor.

The Client_IP factor is a default factor.

1. In Identities page, select the HIGHLY SECURE INTERNAL NETWORK identity and then select **Edit**.
2. In the Add New Identity and Mapping window, select the **Map Identity** subpage.
3. Select the **Map Identity** tab, and then select **Add Mapping**.
4. In the Add New Identity Mapping page, enter the following information:
   - **Child Factor:** Select **Client_IP** to be the child factor.
   - **Operator:** Select **Equal**.
   - **Min Value:** Enter the IP address for the Virtual Machine (for example, `192.0.2.12`). (This is the computer that user `mwaldron` uses. For this tutorial, you can enter the IP address of your own computer. If you are using Microsoft Windows, use the IP address assigned to the Loopback Adapter.)
   - **Max Value:** Leave this field empty.
5. Click **OK**, and then click **OK** again to return to the Identities page.
6. Create the following two identity maps for the NOT SECURE identity, by editing this identity:

Child Factor	Operator	Min Value	Max Value
`Client_IP`	`Less`	`192.0.2.5`	(Leave blank)
`Client_IP`	`Greater`	`192.0.2.20`	(Leave blank)

   The identity maps in the NOT SECURE identity are in a range of IP addresses outside the IP address that user `mwaldron` uses (192.0.2.12). The IP addresses here must be in any range *outside* `mwaldron`'s IP address.

   This identity mapping creates the following condition: If the user logs in from the correct IP address, then Oracle Database Vault decides that the connection is secure, through the HIGHLY SECURE INTERNAL NETWORK identity. However, if the user logs in from an IP address that is less than 192.0.2.5 or greater than 192.0.2.20, then the connection is deemed not secure, through the NO SECURE identity.
7. Click **OK**.
8. Click **Done**, and then click **Finish**.
9. Test the factor identities.

   First, in SQL*Plus, connect as user `mwaldron` but do not specify a database instance.

```
CONNECT mwaldron -- Or, CONNECT mwaldron@hrpdb
Enter password: password

SELECT DVF.F$CLIENT_IP FROM DUAL;
```

The following output should appear:

```
F$CLIENT_IP
-----------------------------------
```

Next:

```
SELECT DVF.F$DOMAIN FROM DUAL;
```

The following output should appear:

```
F$DOMAIN
-----------------------------------
NOT SECURE
```

Because user `mwaldron` is not connecting directly to the database instance, Oracle Database Vault does not recognize the IP address from which he is connecting. In this case, Oracle Database uses the IPC protocol to perform the connection, which sets the IP value to null. Therefore, the identity for this connection is set to NOT SECURE.

Now connect to SQL*Plus by specifying the database instance (for example, `orcl`), and then check the factor identities again:

```
CONNECT mwaldron@orcl
Enter password: password

SELECT DVF.F$CLIENT_IP FROM DUAL;
```

The following output should appear:

```
F$CLIENT_IP
-----------------------------------
192.0.2.12
```

Next:

```
SELECT DVF.F$DOMAIN FROM DUAL;
```

The following output should appear:

```
F$DOMAIN
-----------------------------------
HIGHLY SECURE INTERNAL NETWORK
```

Now that user `mwaldron` is connecting to the `orcl` database instance, his IP address is recognized. This is because the database uses the TCP protocol, so now the host IP value can be populated appropriately. Because the IP address is within the correct range, the factor identity is set to HIGHLY SECURE INTERNAL NETWORK.

Step 4: Create a Rule Set to Set the Hours and Select the Factor Identity

You must create a rule set to work with the factor that you modified.

1. In the Administration page, under Database Vault Components, select **Rule Sets**.
2. In the Rule Sets page, select **Create**.
3. In the Create Rule Set page, enter the following settings:

- **Name:** Enter `Internal DBA Standard Working Hours`.
- **Status:** Select **Enabled**.
- **Evaluation Options:** Select **All True**.

Leave the remaining settings at their defaults.

4. Click **Next** to display the Associate with Rule page.
5. Select **Create Rule**.
6. In the Create Rule window, enter the following settings:
   - **Name:** `Internal DBA`
   - **Expression:** `DVF.F$SESSION_USER='MWALDRON'`

     (When you create an expression with a user name, enter the user name in upper case letters, because that is how the database stores user names.)
7. Click **OK**.
8. Use the **Create Rule** page to create the following additional rules:
   - **Name:** `Internal Network Only`

     Rule Expression: `DVF.F$DOMAIN='HIGHLY SECURE INTERNAL NETWORK'`
   - **Name:** `Week Day`

     Rule Expression: `TO_CHAR(SYSDATE, 'D') BETWEEN '2' AND '6'`
   - **Name:** `Week Working Day Hours`

     Rule Expression: `TO_CHAR(SYSDATE, 'HH24') BETWEEN '08' AND '19'`
9. Click **Done**, and then click **Finish**.

Step 5: Create a Command Rule That Uses the Rule Set

You must create a command rule that uses the rule set that you created.

1. In the Administration page, select **Command Rules**.
2. In the Command Rules page, select **Create**.
3. In the Create Command Rule page, enter the following settings:
   - **Command**: Select **CREATE TABLE** from the list.
   - **Status:** Select **Enabled**.
   - **Applicable Object Owner:** Ensure it is set to **%** (the default).
   - **Applicable Object Name:** Ensure it is set to **%** (the default).
   - **Evaluating Rule Set**: Select **Internal DBA Standard Working Hours** from the list.
4. Click **OK**.

Step 6: Test the Factor Identity Settings

Test the settings by resetting the system clock, logging in as the `mwaldron` administrative user, and then trying to create a table.

1. Set the system time to 9 p.m.

   UNIX: Log in as root and use the `date` command to set the time. For example, assuming the date today is August 15, 2013, you would enter the following:

   ```
   su root
   Password: password

   date --set="15 AUG 2013 21:00:00"
   ```

   Windows: Double-click the clock icon, which is typically at the lower right corner of the screen. In the Date and Time Properties window, set the time to 9 p.m., and then click **OK**.

2. In SQL*Plus, connect as user `mwaldron` and try to create a table. In the following, replace `orcl` with the name of your database instance.

   ```
   CONNECT mwaldron@orcl
   Enter password: password

   CREATE TABLE TEST (num number);
   ```

   The following output should appear:

   ```
   ORA-47400: Command Rule violation for create table on MWALDRON.TEST
   ```

   Because user `mwaldron` is create a table outside working hours, Database Vault prevents him.

3. Reset the system time back to the local time.

4. In SQL*Plus, as user `mwaldron`, try to create the table again.

   ```
   CREATE TABLE TEST (num number);

   Table created.

   DROP TABLE TEST;
   Table dropped.
   ```

   Now that user `mwaldron` is working during his local hours and from the IP address associated with the HIGHLY SECURE INTERNAL NETWORK identity, he can create tables.

5. Reconnect as user `mwaldron` but without adding the database instance name to the connection command, and then try to create the table again.

   ```
   CONNECT mwaldron -- Or, CONNECT mwaldron@hrpdb
   Enter password: password

   CREATE TABLE TEST (num number);
   ```

   The following output should appear:

   ```
   ORA-47400: Command Rule violation for create table on MWALDRON.TEST
   ```

   Even though user `mwaldron` is trying to create a table during the correct time, he cannot because is not directly logged in to the `orcl` database instance. Oracle Database Vault deems him to be using the NOT SECURE identity, and then denies him access.

Step 7: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. Log into the database instance as the `DV_ACCTMGR` user and drop user `mwaldron`.

   ```
   sqlplus bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
   Enter password: password

   DROP USER mwaldron CASCADE;
   ```

2. Remove the CREATE TABLE command rule.

   Return the Administration page and select **Command Rules**. Select the CREATE TABLE command rule and then click **Delete**. In the Confirmation window, select **Yes**.

3. Remove the Internal DBA Standard Working Hours rule set.

   Select **Rule Sets** in the Administration page. In the Rule Sets page, select the Internal DBA Standard Working Hours rule set, and then select **Delete**. In the Confirmation window, select the **Remove rules associated with the rule set** check box, and then select **Yes**.

4. Remove the rules that were associated with the Internal DBA Standard Working Hours rule set.

   Select **Rules** in the Administration page. In the Rules page, select the Internal DBA, Internal Network Only, Week Day, and Week Day Working Hours rules, and then select **Delete**. Select Yes in the Confirmation window.

5. Remove the HIGHLY SECURE INTERNAL NETWORK and NOT SECURE factor identities from the Domain factor.

   In the Administration page and select **Factors**. Select the Domain factor, select **Edit**. Click **Next** until you reach the Identities page. Select the HIGHLY SECURE INTERNAL NETWORK and NOT SECURE factor identities and click **Remove** to remove each one. (Hold the **Control** key down to select multiple items.) In the Confirmation window, select **Yes**. Click **Done**, and then click **Finish**.

Guidelines for Designing Factors

Oracle provides guidelines for designing factors.

- You can use the Oracle utility packages such as `UTL_TCP`, `UTL_HTTP`, `DBMS_LDAP`, and `DBMS_PIPE` to integrate security or other contextual information about the session from external systems.
- Do not specify a retrieval method if the factor identification is set to **Identified By Factors**. Retrieval methods are only needed if you set the factor to **By Method** or **By Constant**.
- Consider using a validation method if a factor has an assignment rule set. Doing so helps to verify that invalid identities are not submitted.
- Use the client-supplied factors such as Program, OS User, and others with caution, because the values that are supplied can only be trusted when the client software is trusted and the communications channel from the client software is known to be secure.

- Only specify an evaluation option of **By Access** if the value returned by the retrieval method could change from one invocation to the next in the same session (for example, time-based factors).
- Optimize the internal logic of a function used for the factor retrieval method using traditional SQL and PL/SQL optimization techniques. For more information about performance and optimization, see *Oracle Database SQL Tuning Guide*.
- If the discrete values returned by the retrieval method are known, be sure to define identities for each value so that you can assign trust levels for them. Trust levels add value to factors as you also can use the trust level in application logic based on factors.
- A security policy based on more factors is generally considered stronger than one based on fewer factors. You can create a new factor that is identified by other factors to store combinations of factors into logical grouping using identity maps. This also makes it easier to label the parent factor when you integrate the factors with the Oracle Label Security labels. (See Integrating Oracle Database Vault with Oracle Label Security for more information.)
- It is generally easier to configure and debug a factor that is labeled **By Self** than one labeled **By Factors** when integrating the Oracle Label Security.
- You can design a database client application to pass one or more security, end-user, or environmental attributes so that they are available to an associated database session. To do this, create a single factor for each attribute and then use an assignment rule set to control when these attributes can be assigned (for example only when using a specific Web application on specified named application server computers). Oracle Database Vault factors used in this fashion are very much like the Oracle procedure `DBMS_SESSION.SET_IDENTIFIER` but also include a capability to control when they can be set. For more information about the `DBMS_SESSION` package, see *Oracle Database PL/SQL Packages and Types Reference* .

How Factors Affect Performance

The complexity of factors affects the performance of your Oracle database instance.

Each factor has elements that are processed, such as its validation method, trust level, and so on. For factors that are evaluated by the session, such as Database_Hostname and `Proxy_User`, Oracle Database Vault performs this processing during session initialization, and then caches the results for subsequent requests for that value.

The default factors listed in Default Factors are cached because they are likely candidates for a typical security policy. However, if you only use five factors (for example, in rule sets or other components), then the other factors consume resources that could otherwise be used elsewhere. In this case, you should remove the unnecessary factors by deleting them. (Oracle Database Vault does not use any of these factors internally, so you can remove them if you do not need them.)

If you have a large number of users or if your application server frequently must create and destroy connections, the resources used can affect system performance. You can delete the unnecessary factors.

You can check system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Cloud Control, which is installed by default with Oracle Database), Automatic Workload Repository (AWR), and `TKPROF`.

See Also:

- *Oracle Database Performance Tuning Guide* to learn how to monitor database performance
- *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements

Factor Related Reports and Data Dictionary Views

Oracle Database Vault provides reports and data dictionary views that display information about factors and their identities.

Table 8-1 lists the Oracle Database Vault reports. See Oracle Database Vault Reports, for information about how to run these reports.

Table 8-1 Reports Related to Factors and Their Identities

Report	Description
Factor Audit Report	Audits factors (for example, to find factors that failed to be evaluated)
Factor Configuration Issues Report	Lists configuration issues, such as disabled or incomplete rule sets, or to audit issues that may affect the factor
Factor Without Identities Report	Lists factors that have had no identities assigned yet
Identity Configuration Issues Report	Lists factors that have invalid label identities or no map for the identity
Rule Set Configuration Issues Report	Lists rule sets that have no rules defined or enabled, which may affect the factors that use them

Table 8-2 lists data dictionary views that provide information about existing factors and factor identities.

Table 8-2 Data Dictionary Views Used for Factors and Factor Identities

Data Dictionary View	Description
DBA_DV_FACTOR View	Lists the existing factors in the current database instance
DBA_DV_FACTOR_LINK View	Shows the relationships of each factor whose identity is determined by the association of child factors
DBA_DV_FACTOR_TYPE View	Lists the names and descriptions of factor types used in the system
DBA_DV_IDENTITY View	Lists the identities for each factor
DBA_DV_IDENTITY_MAP View	Lists the mappings for each factor identity

9

Configuring Secure Application Roles for Oracle Database Vault

Secure application roles enable you to control how much access users have to an application.

- What Are Secure Application Roles in Oracle Database Vault?
  In Oracle Database Vault, you can create a secure application role that you enable with an Oracle Database Vault rule set.
- Creating an Oracle Database Vault Secure Application Role
  You can create a Database Vault secure application role in Database Vault Administrator.
- Modifications to a Secure Application Role
  You can modify an existing secure application role only if it has been created in Oracle Database Vault.
- Security for Oracle Database Vault Secure Application Roles
  Users who have database administrative privileges may try to use the `DROP ROLE` statement to delete Oracle Database Vault secure application roles.
- Deleting an Oracle Database Vault Secure Application Role
  You can delete Oracle Database Vault secure application roles in Oracle Database Vault Administrator.
- How Oracle Database Vault Secure Application Roles Work
  The process flow for an Oracle Database Vault secure application role begins after you create and set the secure application role.
- Tutorial: Granting Access with Database Vault Secure Application Roles
  This tutorial demonstrates how to create a secure application role to control user access to the `OE.ORDERS` table during work hours.
- How Secure Application Roles Affect Performance
  You can check system performance by Oracle Enterprise Manager Cloud Control.
- Secure Application Role Related Reports and Data Dictionary View
  Oracle Database Vault provides reports and a data dictionary view that you can use to analyze Oracle Database Vault secure application roles.

What Are Secure Application Roles in Oracle Database Vault?

In Oracle Database Vault, you can create a secure application role that you enable with an Oracle Database Vault rule set.

Regular Oracle Database secure application roles are enabled by custom PL/SQL procedures. You use secure application roles to prevent users from accessing data from outside an application. This forces users to work within the framework of the application privileges that have been granted to the role.

In a multitenant environment, you only can create a secure application role in a PDB, not in the CDB root or the application root.

The advantage of basing database access for a role on a rule set is that you can store database security policies in one central place, as opposed to storing them in all your applications. Basing the role on a rule set provides a consistent and flexible method to enforce the security policies that the role provides. In this way, if you must update the security policy for the application role, you do it in one place, the rule set. Furthermore, no matter how the user connects to the database, the result is the same, because the rule set is bound to the role. All you need to do is to create the role and then associate it with a rule set. The associated rule set validates the user who is trying to enable the role.

Related Topics

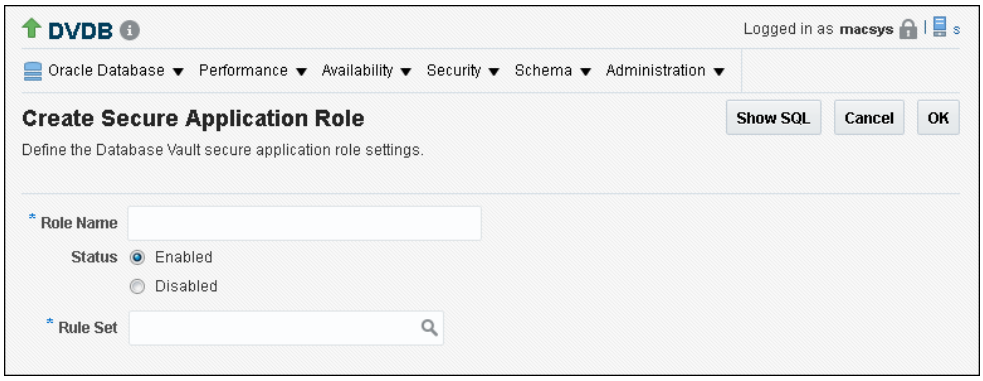
- Oracle Database Vault Secure Application Role APIs
  The `DBMS_MACADM` and `DBMS_MACSEC_ROLES` PL/SQL packages manage Database Vault secure application roles.

Creating an Oracle Database Vault Secure Application Role

You can create a Database Vault secure application role in Database Vault Administrator.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. Create a rule set that contains at least one rule to set the conditions for allowing or disallowing the user to enable the role.

   When you create the underlying rule for the rule set, remember that the rule should validate the user who is trying to enable the role.
3. In the Administration page, under Database Vault Components, click **Secure Application Roles**.
4. In the Secure Application Role page, click **Create**.



5. In the Create Secure Application Role page, enter the following settings:
   - **Role Name:** Enter the name using no more than 30 characters, with no spaces. Ensure that this name follows the standard Oracle naming

conventions for role creation using the `CREATE ROLE` statement, described in *Oracle Database SQL Language Reference*. This attribute is mandatory.

- **Status:** Select either **Enabled** or **Disabled** to enable or disable the secure application role during run time. This attribute is mandatory.
  - **Enabled:** Enables the role to be available for use. That is, users are allowed to call the `DBMS_MACSEC_ROLES.SET_ROLE` function to try to enable the role. Note that whether or not the role will be enabled depends on the evaluation result of the associated rule set.
  - **Disabled:** Disables the role from being available for use. The `DBMS_MACSEC_ROLES.SET_ROLE` function will not be able to enable the role.
- **Rule Set:** From the list, select the rule set that you want to associate with the secure application role. This attribute is mandatory.

  When calling `DBMS_MACSEC_ROLES.SET_ROLE`, if the rule set evaluates to true, then Oracle Database Vault enables the role for the database session. If the rule set evaluates to false, then the role is not enabled.

6. Click **OK**.

Related Topics

- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.
- SET_ROLE Procedure
  The `SET_ROLE` procedure issues the `SET ROLE` PL/SQL statement for specified roles.
- Propagating Oracle Database Vault Configurations to Other Databases
  You can propagate Database Vault configurations (such as a realm configuration) to other Database Vault-protected databases.

Modifications to a Secure Application Role

You can modify an existing secure application role only if it has been created in Oracle Database Vault.

You cannot modify secure application roles or database roles that have been created outside of Oracle Database Vault. If you want to modify an existing Oracle Database role so that it can work with Oracle Database Vault, create a new secure application role in Oracle Database Vault and then grant the existing role to the secure application role. For example, in SQL*Plus:

```
GRANT myExistingDBrole TO myDVrole;
```

After you create a new secure application role, you must modify your code to use this new role. You can use `DBMS_MACSEC_ROLES.SET_ROLE` in your application code to accomplish this.

Related Topics

- SET_ROLE Procedure
  The `SET_ROLE` procedure issues the `SET ROLE` PL/SQL statement for specified roles.

Security for Oracle Database Vault Secure Application Roles

Users who have database administrative privileges may try to use the `DROP ROLE` statement to delete Oracle Database Vault secure application roles.

Whenever an Oracle Database Vault secure application role has been created, Database Vault adds the secure application role to the Oracle Database Vault realm. This prevents database administrator from deleting the secure application role using the `DROP ROLE` statement.

Deleting an Oracle Database Vault Secure Application Role

You can delete Oracle Database Vault secure application roles in Oracle Database Vault Administrator.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. If necessary, locate the various references to the secure application roles by querying the role-related Oracle Database Vault views.
3. Check and modify any applications that may be using the secure application role that you want to delete.
4. In the Administration page, under Database Vault Components, click **Secure Application Roles**.
5. In the Secure Application Roles page, select the role that you want to remove.
6. Click **Delete**.
7. In the Confirmation window, click **Yes**.

Related Topics

- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

How Oracle Database Vault Secure Application Roles Work

The process flow for an Oracle Database Vault secure application role begins after you create and set the secure application role.

1. Create or update the role either in Oracle Database Vault Administrator or by using the secure application role-specific functions in the `DBMS_MACADM` package.

   See DBMS_MACADM Secure Application Role Procedures for more information.
2. Modify your application to call the role, by using the `DBMS_MACSEC_ROLES.SET_ROLE` function.

   See SET_ROLE Procedure for more information.

3. Oracle Database Vault then evaluates the rule set associated with the secure application role.

   If the rule set evaluates to true, then Oracle Database Vault enables the role for the current session. If the rule set evaluates to false, the role is not enabled. In either case, Oracle Database Vault processes the associated auditing and custom event handlers for the rule set associated with the secure application role.

Tutorial: Granting Access with Database Vault Secure Application Roles

This tutorial demonstrates how to create a secure application role to control user access to the `OE.ORDERS` table during work hours.

- About This Tutorial
  In this tutorial, you restrict the `SELECT` statement on the `ORDERS` table in the `OE` schema to a specific set of users.
- Step 1: Create Users for This Tutorial
  First, you must create users for the tutorial.
- Step 2: Enable the OE User Account
  The `OE` schema will be used for this tutorial.
- Step 3: Create the Rule Set and Its Rules
  The rule set and rules will restrict who can modify orders in the `OE.ORDERS` table.
- Step 4: Create the Database Vault Secure Application Role
  The Database Vault secure application role will be set when the rule set conditions are satisfied.
- Step 5: Grant the SELECT Privilege to the Secure Application Role
  The secure application role must be granted the `SELECT` privilege.
- Step 6: Test the Database Vault Secure Application Role
  With all the components in place, you can test the Database Vault secure application role.
- Step 7: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

About This Tutorial

In this tutorial, you restrict the `SELECT` statement on the `ORDERS` table in the `OE` schema to a specific set of users.

Furthermore, these users can only perform these statements on the `OE.ORDERS` table from within the office, not from a remote connection. To accomplish this, you create an Oracle Database Vault secure application role that is enabled for the user only if the user passes the checks enforced by the rule set that you associate with the secure application role.

Step 1: Create Users for This Tutorial

First, you must create users for the tutorial.

1. Log in to SQL*Plus as a user who has been granted the DV_ACCTMGR role.

   For example:

   ```
   sqlplus bea_dvacctmgr
   Enter password: password
   ```

   In a multitenant environment, you must connect to the appropriate pluggable database (PDB).

   For example:

   ```
   sqlplus bea_dvacctmgr@hrpdb
   Enter password: password
   ```

   To find the available PDBs, query the DBA_PDBS data dictionary view. To check the current PDB, run the show con_name command.

2. Create the following user accounts:

   ```
   GRANT CREATE SESSION TO eabel IDENTIFIED BY password;
   GRANT CREATE SESSION TO ahutton IDENTIFIED BY password;
   GRANT CREATE SESSION TO ldoran IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

Step 2: Enable the OE User Account

The OE schema will be used for this tutorial.

1. In SQL*Plus, connect as the DV_ACCTMGR user.

   For example:

   ```
   CONNECT bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
   Enter password: password
   ```

2. Check the account status of the OE account.

   ```
   SELECT USERNAME, ACCOUNT_STATUS FROM DBA_USERS WHERE USERNAME = 'OE';
   ```

3. If the OE account is locked and expired, unlock it and assign it a new password.

   ```
   ALTER USER OE ACCOUNT UNLOCK IDENTIFIED BY password;
   ```

Step 3: Create the Rule Set and Its Rules

The rule set and rules will restrict who can modify orders in the OE.ORDERS table.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the DV_OWNER or DV_ADMIN role and the SELECT ANY DICTIONARY privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.

2. In the Administration page, select **Rule Sets**.

   The Rule Sets page appears.

3. Click **Create**.

   The Create Rule Set page appears.

4. Enter the following information:

- **Name:** Enter `Can Modify Orders`.
- **Description:** Enter `Rule set to control who can modify orders in the OE.ORDERS table`.
- **Status:** Select **Enabled**.
- **Evaluation Options:** Select **All True.**

5. Leave the remaining settings and their defaults, and then click **Next** to go to the Associate with Rules page.
6. Click **Create Rule** and in the Create Rule dialog box, enter the following settings:
   - **Name:** `Check IP Address`
   - **Expression:** `DVF.F$CLIENT_IP = 'your_IP_address'`

   For the Check IP Address rule, replace *your_IP_address* with the IP address for your computer. In a real-world scenario, you would create an expression that includes all the IP addresses for the users who should be allowed access.

   This rule uses the default factor Client_IP. If this factor has been removed, then you can use the following rule expression instead:

   ```
   UPPER(SYS_CONTEXT('USERENV','IP_ADDRESS')) = 'your_IP_address'
   ```
7. Click **OK**.
8. Click **Create Rule** again and in the Create Rule dialog box, enter the following settings:
   - **Name:** `Check Session User`
   - **Expression:** `DVF.F$SESSION_USER IN ('EABEL','AHUTTON')`

   This rule uses the default factor Session_User. If this factor have been removed or modified, you can use the following rule expression instead:

   ```
   UPPER(SYS_CONTEXT('USERENV','SESSION_USER')) IN ('EABEL','AHUTTON')
   ```
9. Click **OK**.
10. Click **Done**, then click **Finish**.

Step 4: Create the Database Vault Secure Application Role

The Database Vault secure application role will be set when the rule set conditions are satisfied.

1. In Oracle Database Vault, return to the Administration page.
2. Under Administration, select **Secure Application Roles**.

   The Secure Application Roles page appears.
3. Click **Create**.

   The Create Role page appears.
4. In the **Role** box, enter `ORDERS_MGMT` to name the role.
5. Under Rule Set, select **Can Modify Orders**.
6. Click **OK**.

At this stage, the Database Vault secure application role and its associated rule set are created, though the role does not yet have any privileges.

Step 5: Grant the SELECT Privilege to the Secure Application Role

The secure application role must be granted the SELECT privilege.

1. In SQL*Plus, connect as user OE.

```
CONNECT OE -- Or, CONNECT OE@hrpdb
Enter password: password
```

2. Grant the SELECT privilege to the ORDERS_MGMT Database Vault Secure application role.

```
GRANT SELECT ON ORDERS TO ORDERS_MGMT;
```

Step 6: Test the Database Vault Secure Application Role

With all the components in place, you can test the Database Vault secure application role.

1. In SQL*Plus, connect directly to the database as user eabel.

```
connect eabel@orcl
Enter password: password
```

   Replace orcl with the name of your database instance.

2. Set the ORDERS_MGMT role.

```
EXEC DBMS_MACSEC_ROLES.SET_ROLE('ORDERS_MGMT');
```

   Typically, you would embed this call in the application that the user logs in to.

3. Select from the OE.ORDERS table.

```
SELECT COUNT(*) FROM OE.ORDERS;
```

   The following output should appear:

```
  COUNT(*)
----------
       105
```

   Because user eabel is logging directly into the database from the correct IP address and is listed as a valid session user, she can select from the OE.ORDERS table. If user ahutton logs in to SQL*Plus in the same manner, she also can select from the OE.ORDERS table.

4. Reconnect as user eabel without specifying the database instance, and then try to select from the OE.ORDERS table again.

```
CONNECT eabel
Enter password: password

EXEC DBMS_MACSEC_ROLES.SET_ROLE('ORDERS_MGMT');
```

   The following output should appear:

```
Error at line 1:
ORA-47305: Rule Set Violation on SET ROLE (Can Modfiy Orders)
...
```

Next:

```
SELECT COUNT(*) FROM OE.ORDERS;
```

The following output should appear:

```
ERROR at line 1:
ORA-00942: table or view does not exist
```

Even though user `eabel` is a valid user, she has violated the Check IP Address rule in the rule set, so she cannot enable the `ORDERS_MGMT` role. The only way for the IP address to be recognized is to connect by specifying the database instance, as user `eabel` did in Step 1. (For an explanation about how this works, see Step 9 in Step 3: Map the Domain Factor Identities to the Client_IP Factor, in Configuring Factors.)

5. Connect as user `ldoran`.

```
CONNECT ldoran -- Or, CONNECT ldoran@hrpdb
Enter password: password
```

6. Enter the following statements:

```
EXEC DBMS_MACSEC_ROLES.SET_ROLE('ORDERS_MGMT');
SELECT COUNT(*) FROM OE.ORDERS;
```

Because user `ldoran` is not a valid user, she cannot enable the `ORDERS_MGMT` role. Therefore, she cannot select from the `OE.ORDERS` table.

Step 7: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. Delete the `ORDERS_MGMT` secure application role: From the Secure Application Roles page, select the `ORDERS_MGMT` secure application role, and then click **Delete**, and then **Yes** in the Confirmation dialog box.
3. Select the Rule Sets page, select the Can Modify Orders rule set, and then click **Delete**.
4. In the Confirmation dialog box, select **Yes** to remove the rule set.
5. Select the Rules page, select the Check IP Address and Check Session User rules, and then select **Delete**. Select **Yes** in the Confirmation box.

   Hold the **Control** key down to select multiple rules.
6. In SQL*Plus, connect as the Database Vault Account Manager and drop the users.

   For example:

```
CONNECT bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
Enter password: password

DROP USER eabel;
```

```
DROP USER ahutton;
DROP USER ldoran;
```

7. If unnecessary, lock and expire the OE user account.

```
ALTER USER OE ACCOUNT LOCK PASSWORD EXPIRE;
```

How Secure Application Roles Affect Performance

You can check system performance by Oracle Enterprise Manager Cloud Control.

Other tools that you can use are Automatic Workload Repository (AWR) and TKPROF.

> **See Also:**
>
> - *Oracle Database Performance Tuning Guide* to learn how to monitor database performance
> - *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements

Secure Application Role Related Reports and Data Dictionary View

Oracle Database Vault provides reports and a data dictionary view that you can use to analyze Oracle Database Vault secure application roles.

Table 9-1 lists the Oracle Database Vault reports. See Oracle Database Vault Reports for information about how to run these reports.

Table 9-1 Reports Related to Secure Application Roles

Report	Description
Secure Application Role Audit Report	Lists audit records generated by the Oracle Database Vault secure application role-enabling operation. To generate this type of audit record, enable auditing for the rule set associated with the role.
Secure Application Configuration Issues Report	Lists secure application roles that have nonexistent database roles, or incomplete or disabled rule sets
Rule Set Configuration Issues Report	Lists rule sets that have no rules defined or enabled, which may affect the secure application roles that use them
Powerful Database Accounts and Roles Reports	Provides information about powerful database accounts and roles

You can use the `DBA_DV_ROLE` data dictionary view to find the Oracle Database Vault secure application roles used in privilege management. See DBA_DV_ROLE View for more information.

10

Configuring Oracle Database Vault Policies

You can use Oracle Database Vault policies to implement frequently used realm and command rule settings.

- What Are Database Vault Policies?
  An Oracle Database Vault policy groups local realms and command rules into a named policy that you can enable or disable as necessary.
- Default Oracle Database Vault Policies
  Oracle Database Vault provides two default policies that you can use to better secure user accounts and system privileges.
- Creating an Oracle Database Policy
  To create an Oracle Database Vault policy, you create a container policy that specifies the realms and command rules that encompass the policy.
- Modifying an Oracle Database Vault Policy
  You can use Enterprise Manager Cloud Control to modify an Oracle Database Vault policy.
- Deleting an Oracle Database Vault Policy
  You can use Enterprise Manager Cloud Control to delete Oracle Database Vault policies.
- Related Data Dictionary Views
  Oracle Database Vault provides data dictionary views that are useful for analyzing Database Vault policies.

What Are Database Vault Policies?

An Oracle Database Vault policy groups local realms and command rules into a named policy that you can enable or disable as necessary.

- About Oracle Database Vault Policies
  Oracle Database Vault policies can group realm and command rule definitions into one policy, which then can be collectively enabled or disabled.
- Oracle Database Vault Policies in a Multitenant Environment
  Oracle Database Vault policies are only local to the pluggable database (PDB) in which they were created.

About Oracle Database Vault Policies

Oracle Database Vault policies can group realm and command rule definitions into one policy, which then can be collectively enabled or disabled.

Database Vault policies enable you to delegate limited realm administration privileges to database users without giving them the powerful privileges that the `DVADM` and `DVOWNER` roles provide. Oracle Database Vault provides default policies.

For example, suppose you have a set of Oracle Database Vault objects that are related to a particular application, such as a realm and several command rules. You

can use a Database Vault policy to group these objects into one policy. You then can designate a policy administrator to manage adding users to a realm for this application and for enabling or disabling the policy. If there is only one primary application, then it can be used for manageability where a user can enable, disable, or simulate (use simulation mode) all related objects with one command rather than issuing a command for each included Database Vault object.

How the enablement of the individual realms and command rules works depends on how you set the policy state of the policy, as follows:

- Full enabled mode (`DBMS_MACADM.G_ENABLED`) sets the policy to take precedence over the individual enablement settings of the associated realms and command rules. For example, if the associated objects of a policy are individually disabled, then they will be enabled if the policy is enabled. (Conversely, you can set `DBMS_MACADM.G_PARTIAL` to allow the embedded security objects to set their own enabled, disabled, or simulation mode.)
- Partial enabled mode (`DBMS_MACADM.G_PARTIAL`) enables the associated realms and command rules to have different status settings (`ENABLED`, `DISABLED`, and `SIMULATION`). The other policy status choices force all associated controls to the same status dictated by the policy. Setting the policy status to partial allows each realm and command rule to change status as required.
- Simulation mode (`DBMS_MACACM.G.SIMULATION`) enables the policy but writes violations to realms or command rules to a designated log table with information about the type of violation, such as a user name or the SQL statement that was used. Simulation forces every security object in the policy to be in simulation mode.
- Disabled mode (`DBMS_MACADM.G_DISABLED`) disables the policy after you create it.

In general, to create a Database Vault policy, you perform the following steps:

1. Create the necessary realms and command rules to use in the policy.
2. Create the Database Vault policy.

   You can use the `DBMS_MACADM.CREATE_POLICY` procedure to create the policy.
3. Add one or more realms to the policy.

   You can use the `DBMS_MACADM.ADD_REALM_TO_POLICY` procedure to add realms to the policy.
4. Add one or more command rules to the policy.

   You can use the `DBMS_MACADM.ADD_CMD_TO_POLICY` procedure to add command rules to the policy.
5. Add one or more database users as owners of the policy.

   You can use the `DBMS_MACADM.ADD_OWNER_TO_POLICY` procedure to add users to the policy. Afterward, grant this user the `DV_POLICY_OWNER` role. This user will be able to perform a limited set of tasks: changing the policy state, adding or removing authorization from a realm, and having the `SELECT` privilege for a set of the `DVSYS.POLICY_OWNER*` data dictionary views. By default, the `DVOWNER` user owns the policy.

After the policy is created, it can be used right away.

This section explains how to configure policies by using the Oracle Database Vault Administrator pages in Oracle Enterprise Manager Cloud Control. To configure policies

by using the PL/SQL interfaces and packages provided by Oracle Database Vault, you must use the DBMS_MACADM PL/SQL package.

Related Topics

- Default Oracle Database Vault Policies
  Oracle Database Vault provides two default policies that you can use to better secure user accounts and system privileges.
- Oracle Database Vault Policy APIs
  You can use the DBMS_MACADM PL/SQL package to manage Oracle Database Vault policies.
- DV_POLICY_OWNER Database Vault Owner Role
  The DV_POLICY_OWNER role enables database users to manage to a limited degree Oracle Database Vault policies.

Oracle Database Vault Policies in a Multitenant Environment

Oracle Database Vault policies are only local to the pluggable database (PDB) in which they were created.

That is, if you created the policy in a PDB, then only local realms and command rules can be added to it. You cannot create Database Vault policies that can have common realms or common command rules.

Default Oracle Database Vault Policies

Oracle Database Vault provides two default policies that you can use to better secure user accounts and system privileges.

You can use the default policies in your own security configurations. If you do not need them, then you can remove them because they are not needed for internal use by Oracle Database Vault.

The default policies are as follows:

- Oracle Account Management Controls enforces controls over user-related operations within Oracle Database Vault. It is used to prevent ad hoc user account creation, user deletions, and other user account-related operations by unauthorized privileged users. It includes the Database Vault Account Management realm and user account management command rules for SQL statements such as CREATE USER.
- Oracle System Protection Controls enforces controls on important database schemas, privileges, and roles that are associated with the default Oracle Database environment. It includes the realms such as Oracle Default Schema Protection Realm and command rules for the system management SQL statement ALTER SYSTEM.

Related Topics

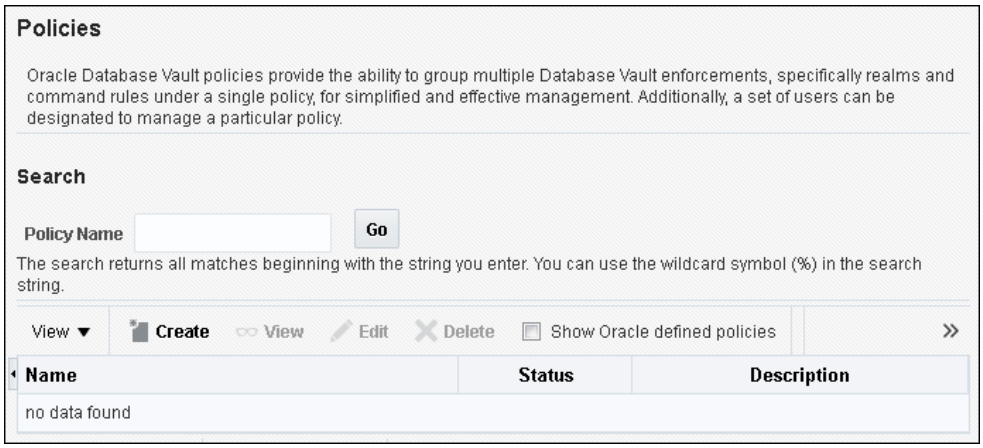
- DBA_DV_POLICY_OBJECT View
  The DBA_DV_POLICY_OBJECT data dictionary view lists information about the objects that are protected by Oracle Database Vault policies in the current database instance.

Creating an Oracle Database Policy

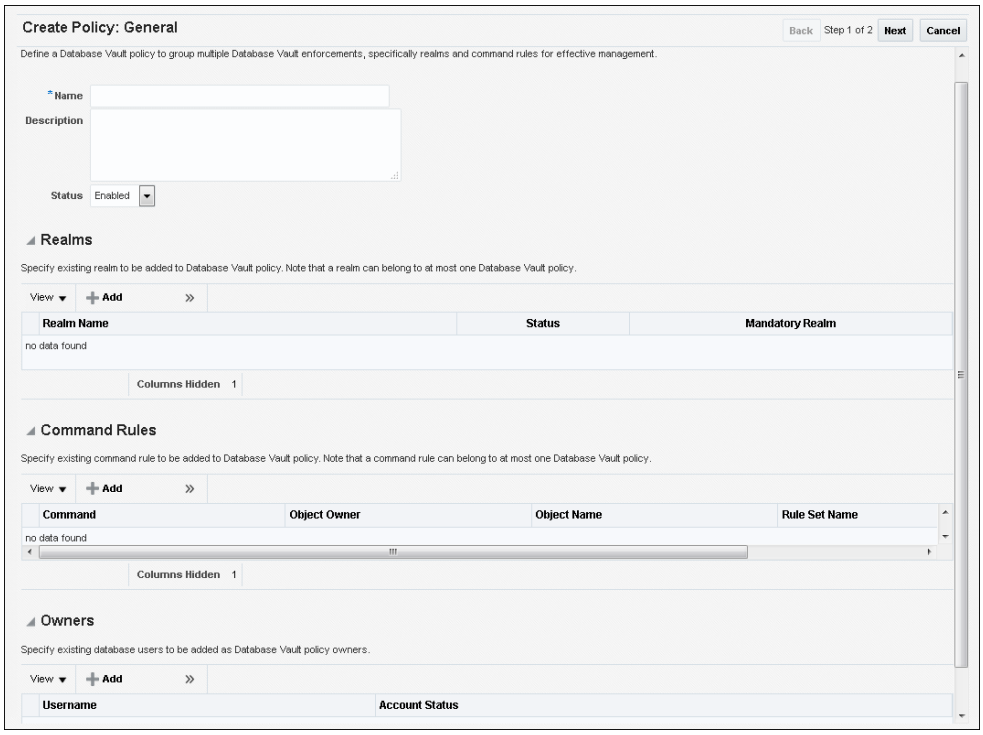
To create an Oracle Database Vault policy, you create a container policy that specifies the realms and command rules that encompass the policy.

You can enable the policy during creation time, or enable it later on.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. Create the realms and command rules that you want to associate with the policy, using Creating a Realm and Creating a Rule Set.
3. In the Administration page, under Database Vault Components, click **Policies** to display the Policies page.



4. In the Policies page, click **Create** to display the Create Policy page.



5. In the Create Policy page, under General, enter the following settings:
   - **Name:** Enter a policy name, up to 128 characters.
   - **Description:** Enter a description of the policy, up to 4000 characters.
   - **Status:** Select from the following:
     - **Enabled** enables the policy after you create it.
     - **Disabled** disables the policy after you create it.
     - **Simulation** sets the policy to simulation mode. In simulation mode, any violations to realms or command rules used in the policy are logged in a designated log table with sufficient information to describe the error, such as the user name or SQL statement used.
     - **Partial** enables the enforcement state of realms or command rules associated with the policy to be changed individually.
6. Under Realms, click **Add** to select a realm to add to the policy. Then click **OK**.
7. Under Command Rules, click **Add** to select a command rule to add to the policy. Then click **OK**.
8. Under Owners, click **Add** to add an owner to the policy. Then click **OK**.
9. Click **Next**.
10. In the Review page, click **Finish**.
11. So that the Database Vault policy owner can query policy related views and execute the allowed procedures, grant this user the `DV_POLICY_OWNER` role.

    For example:

    ```
    GRANT DV_POLICY_OWNER TO psmith;
    ```

Modifying an Oracle Database Vault Policy

You can use Enterprise Manager Cloud Control to modify an Oracle Database Vault policy.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Policies**.
3. Select the row for the policy that you want to change.
4. Click **Edit**.
5. In the Edit Policy page, modify the settings as necessary.
6. Click **Next**, and then click **Finish**.

Deleting an Oracle Database Vault Policy

You can use Enterprise Manager Cloud Control to delete Oracle Database Vault policies.

When you delete an Oracle Database Vault policy, the underlying realms and command rules are preserved, and they retain their individual enablement status.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Administration page, under Database Vault Components, click **Policies**.
3. Select the row for the policy that you want to delete, click **Delete**, and then click **Yes** in the confirmation dialog box.

Related Data Dictionary Views

Oracle Database Vault provides data dictionary views that are useful for analyzing Database Vault policies.

Table 10-1 lists data dictionary views that provide information about existing Oracle Database Vault policies.

Table 10-1 Data Dictionary Views Used for Oracle Database Vault Policies

Data Dictionary View	Description
DBA_DV_POLICY View	Lists the Database Vault policies, a description, and their state
DBA_DV_POLICY_OBJECT View	Provides detailed information about the policies, such as the associated realms and command rules
DBA_DV_POLICY_OWNER View	Lists the owners of Database Vault policies

Table 10-1 (Cont.) Data Dictionary Views Used for Oracle Database Vault Policies

Data Dictionary View	Description
DBA_DV_REALM_AUTH View	Enables users who have been granted the DV_POLICY_OWNER role to find information about the authorization that was granted to realms that have been associated with Database Vault policies, such as the realm name, grantee, and associated rule set.
DVSYS.POLICY_OWNER_COMMAND_RULE View	Enbles users who have been granted the DV_POLICY_OWNER role to find information about the command rules that have been associated with Database Vault policies, such as the command rule name.
DVSYS.POLICY_OWNER_POLICY View	Enbles users who have been granted the DV_POLICY_OWNER role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners
DVSYS.POLICY_OWNER_POLICY View	Eenables users who have been granted the DV_POLICY_OWNER role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners
DVSYS.POLICY_OWNER_REALM View	Eenables users who have been granted the DV_POLICY_OWNER role to find information about the realms that have been associated with Database Vault policies, such as the realm name, audit options, or type
DVSYS.POLICY_OWNER_REALM_OBJECT View	Enables users who have been granted the DV_POLICY_OWNER role to find information about the objects that have been added to realms that are associated with Database Vault policies, such as the realm name, grantee, and associated rule set
DVSYS.POLICY_OWNER_RULE View	Enables users who have been granted the DV_POLICY_OWNER role to find information about the rules that have been associated with rule sets in Database Vault policies, such as the rule name and its expression
DVSYS.POLICY_OWNER_RULE_SET View	Enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies, such as the rule set name, its handler information, and whether it is enabled

Table 10-1 (Cont.) Data Dictionary Views Used for Oracle Database Vault Policies

Data Dictionary View	Description
DVSYS.POLICY_OWNER_RULE_SET_RULE View	Enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that contain rules used in Database Vault policies, such as the rule set name and whether it is enabled

11

Using Simulation Mode for Logging Realm and Command Rule Activities

Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

- About Simulation Mode
  Simulation mode enables you to capture violations in a simulation log instead of blocking SQL execution by Oracle Database Vault realms and command rules.
- Simulation Mode Use Cases
  Simulation mode is useful for testing a development configuration of new realms and command rules.
- Logging Realms in Simulation Mode
  You can set both regular and mandatory realms in simulation mode.
- Tutorial: Tracking Violations to a Realm Using Simulation Mode
  This tutorial shows how to create a realm that uses simulation mode and then test violations to the realm.
- Use Cases and Guidelines for Logging Realms in Simulation Mode
  Oracle provides a set of use cases and guidelines for logging realms in simulation mode.

About Simulation Mode

Simulation mode enables you to capture violations in a simulation log instead of blocking SQL execution by Oracle Database Vault realms and command rules.

Simulation mode stores the errors that are captured in one location for easy analysis. To use simulation mode, when you create or update a realm or command rule, instead of enabling or disabling the realm or command rule, you can set it to simulation mode. The realm or command rule is still enabled, but because violations are not blocked and are instead recorded to the simulation log file, you can test it for any potential errors before you enable it for a production environment. When simulation mode is enabled, the report may include violations for multiple realms or command rules. For more detailed reports that can help you better identify the source of user SQL statements, you can configure simulation mode to include the PL/SQL call stack. The call stack captures the calling procedures and functions recursively to better troubleshoot the Database Vault audit records. Call stack information is stored in the `PL_SQL_STACK` column in the `DVSYS.DBA_DV_SIMULATION_LOG` data dictionary view.

For example, the following creation statement for a realm enables simulation mode and generates the PL/SQL call stack:

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
 realm_name    => 'HR Apps',
 description   => 'Realm to protect the HR realm',
 enabled       => DBMS_MACUTL.G_SIMULATION,
 audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
```

```
 realm_type    => 1,
 realm_scope   => DBMS_MACUTL.G_SCOPE_LOCAL,
 pl_sql_stack  => TRUE);
END;
/
```

At this stage, SQL statements that violate realms or command rules are still able to execute, but these activities are recorded to the DBA_DV_SIMULATION_LOG data dictionary view. For example, the following query finds violations against the HR Apps realm and any other realms or command rules that have been configured for simulation mode:

```
SELECT USERNAME, COMMAND, SQLTEXT, VIOLATION_TYPE FROM DBA_DV_SIMULATION_LOG WHERE
REALM_NAME = "HR APPS";

USERNAME  COMMAND    SQLTEXT                          VIOLATION_TYPE

--------  ---------- -------------------------------  --------------
DGRANT    SELECT     SELECT SALARY FROM HR.EMPLOYEES; Realm Violation
```

After you have completed testing the realm or command rule, a user who has been granted the DV_ADMIN or DV_OWNER role can clear the DBA_DV_SIMULATION_LOG data dictionary view by deleting the contents of the underlying table of this view, DVSYS.SIMULATION_LOG$.

For example:

```
DELETE FROM DVSYS.SIMULATION_LOG$;
```

Or:

```
DELETE FROM DVSYS.SIMULATION_LOG$ WHERE COMMAND = 'SELECT';
```

Simulation Mode Use Cases

Simulation mode is useful for testing a development configuration of new realms and command rules.

Use cases are as follows:

- Application certification

  When you are certifying applications, you can use simulation mode as follows in an application test environment:

  1. Put all schemas for the application into mandatory realms with simulation mode enabled.
  2. Run a full regression test.
  3. Analyze the simulation mode log by querying the DBA_DV_SIMULATION_LOG data dictionary view to find who can access these schemas.
  4. Update the realms with new authorizations, and the enable the realms (that is, not using simulation mode).
  5. Re-run the regression test.

- Introduction of a new command rule

  You can use simulation mode on a production database that has Oracle Database Vault enabled.

1. Put the new command rule into production in simulation mode for however many weeks that are necessary.
2. Analyze the simulation mode log by querying `DBA_DV_SIMULATION_LOG` to determine if the command rule is working correctly.
3. Make changes to the command rule as necessary.
4. Enable the command rule.

- Putting a new realm into a production database in simulation mode.

  This method can help to find the system context information needed to set the trusted path rules in rule sets and find authorized users for realms.

  1. Create the new realm in mandatory mode and add the protected objects
  2. Do not add any authorized users.
  3. Run applications and development operations from the normal IP addresses that will be used.
  4. Check the simulation log file for both authorized users and system context information that you can use to create trusted paths.
  5. Create the trusted paths, and then add the authorized users.
  6. Clear the simulation log and run the application and development operation tasks again.
  7. After a period of time, review the simulation log. If all the controls were updated correctly, then the simulation log is empty. Log entries in the simulation mode indicate additional changes that you need to make to the realm and rule sets or the log entries may indicate a malicious use.

Logging Realms in Simulation Mode

You can set both regular and mandatory realms in simulation mode.

- Considerations When Logging Realms in Simulation Mode
  There are several use cases to consider if you want to use realms in simulation mode.
- Use Case: All New Realms in Simulation Mode
  In this use case, all realms are either mandatory or regular and all user-created realms are in simulation mode.
- Use Case: New Realms Introduced to Existing Realms
  In this use case, you add a set of new realms to a database that has an existing set of realms.
- Use Case: Testing the Addition of New Objects in a Realm
  In this use case, you add new objects to an existing realm and then test it using simulation mode without removing the current realm protections.
- Use Case: Testing the Removal of Objects from a Realm
  In this use case, you test the removal of objects to an existing realm.
- Use Case: Testing the Addition of an Authorized User to a Realm
  In this use case, you loosen security controls by adding more users. You do not need to simulate anything if you are simply adding more authorized users.

- Use Case: Testing the Removal of an Authorized User from a Realm
  In this use case, you want to drop an authorized user and use simulation mode to check if the user still needs to access the realm.
- Use Case: Testing New Factors with Realms
  In this use case, you want to test changes to factors.
- Use Case: Testing Changes to an Existing Command Rule
  In this use case, you test changes to an existing command rule while keeping the original command rule enabled.

Considerations When Logging Realms in Simulation Mode

There are several use cases to consider if you want to use realms in simulation mode.

- Testing an application with all new Database Vault controls: all realms are in simulation mode
- Adding a realm to existing working Database Vault controls: only a subset of realms are in simulation mode
- Adding new objects to an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Dropping one or more existing objects from an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Adding new authorized users to an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Dropping one or more existing authorized users from an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Adding or changing factors in an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Testing changes to a command rule in production while keeping the original command rule enabled

When a user executes a SQL statement and it fails, it may fail for realms that are enabled, fail for realms that are simulated, or it could fail for both of these reasons. There could be mandatory realms, regular realms, or both. These conditions determine the data that is captured in the simulation log.

After you create the use cases that are described in the next sections, regular realms are completely overpowered by mandatory realms when an object has both types of realms protecting it. In all cases where mandatory and regular realms protect the same object, regular realms can be ignored with regard to simulation logs. Only mandatory realm failures are captured in the simulation logs. The only time regular realm failures are entered into the simulation log is when all realms for an object are regular realms. And then, the following must happen for regular realms to be written to the simulation log:

- All regular realms in simulation mode fails and
- All regular realms that are enabled also fail

If at least one enabled or simulation regular realm succeeds, then no simulation regular realms are logged.

Use Case: All New Realms in Simulation Mode

In this use case, all realms are either mandatory or regular and all user-created realms are in simulation mode.

Examples are as follows:

- Mandatory realms only, which are all in simulation mode
  - The user is authorized to execute the SQL statement in all mandatory realms. Nothing is captured in the simulation log table.
  - The user fails one or more mandatory realm checks. All realm check failures are logged to the simulation log. Mandatory realm checks where the user's SQL statement succeeded is not logged.

    In this example, there are three mandatory realms. The user SQL statement succeeds with one realm and fails with the other two. Only the two failed realm checks are recorded in the simulation log.
- Regular realms only, which are all in simulation mode
  - The user is authorized to execute the SQL statement in at least one regular realm. The user should have access to the data so nothing is logged to the simulation log.
  - The user is not authorized to execute the SQL statement in all regular realms. The simulation log captures all the failed realm authorization failures. This enables the user to select which realm to which the user should be authorized. The SQL only needs to be authorized in one regular realm to work and not all regular realms need to be updated to authorize the SQL.
- Mix of mandatory and regular realms, which are all in simulation mode
  - In this case, you capture the key realms when a user is rejected. In the case with mandatory and regular realms, the mandatory realms are the key realms. All mandatory realms must pass the authorization check for the user to gain access. In fact, regular realms could be considered superfluous when mandatory realms are protecting an object. So in cases where there are both mandatory and regular realms protecting the same object, it is only the mandatory realms that control if the SQL statement is blocked or allowed to execute. It does not matter whether the user was authorized to the regular realm or not. This example follows the rules for the first scenario, for mandatory realms in simulation mode.
  - The user is authorized to execute the SQL statement in all mandatory realms. Nothing is captured in the simulation log table. Even though the user may succeed or fail in one or more regular realms, nothing about regular realm failure is captured.
  - The user fails one or more mandatory realm checks. All realm check failures are logged to the simulation log. Mandatory realm checks where the user SQL statement succeeded are not be logged.

    For example, there are three mandatory realms. The user SQL statement succeeds with one realm and fails with the other two. Only the two failed realm checks are recorded in the simulation log.

    No regular realms need to be captured, because only the mandatory realms need to be captured in the simulation log.

Use Case: New Realms Introduced to Existing Realms

In this use case, you add a set of new realms to a database that has an existing set of realms.

The existing realms are enabled and working. The new realms are in simulation mode. This use case applies only if both simulation and enabled realms are protecting the same object.

Examples:

- New mandatory realms in simulation mode with existing enabled mandatory realms. This use case shows some additional mandatory realms for an object: adding more security to an existing object.
  - Enabled mandatory realms and mandatory realms in simulation mode all successful with user SQL statement: in this case, the SQL executes normally and nothing is captured
  - Enabled mandatory realms (at least one) fails and mandatory realms in simulation mode all successful: SQL is blocked and nothing is written to the simulation log
  - Enabled mandatory realms (at least one) fails and mandatory realms in simulation mode has one or more failures: SQL is blocked and all failing mandatory realms in simulation mode are entered into simulation log
  - Enabled mandatory realms all successful and mandatory realms in simulation mode have at least one failure: SQL is not blocked, all failed mandatory realms in simulation mode entered into simulation log
- New regular realms in simulation mode with existing enabled regular realms: More regular realms are added to a security object, providing new ways for users to access sensitive data
  - Enabled regular realms (at least one) and regular realms in simulation mode (at least one) successful: the user SQL executes normally with nothing written to simulation log
  - Enabled regular realms (at least one) is successful, and regular realms in simulation mode all fail: user SQL executes normally, nothing is entered into the simulation log
  - Enabled regular realms all fail and regular realms in simulation mode all fail: the user SQL is blocked and all regular realms in simulation mode are entered into simulation log. The user must evaluate which regular realm to authorize to if needed. The current implementation blocks the SQL and does not add the regular realms in simulation mode into the simulation log because the enabled regular realm would have blocked it anyway. This must change because the user may have added a new realm to authorize the SQL in this use case. There is no way to tell what happened if the new SQL should have worked, but is blocked by all regular realms in simulation mode as well (when one of the regular realms in simulation mode was designed to allow it to work). This would simulate an entry into the audit log for this situation.
  - Enabled regular realms all fail and regular realms in simulation mode (at least one) successful: the user SQL is blocked and nothing is written to the simulation log.

- New regular realms with existing enabled mandatory realms: You do not need to do anything in this situation. The enabled mandatory realms will continue to control the objects and the new regular realms in simulation mode will have no impact if they are enabled or not. No simulation logs should be generated in this case.
- New mandatory realms in simulation mode with existing enabled regular realm: While the enabled regular realm controls the objects for now, when the new mandatory realms in simulation mode are enabled, then they will have full control over the objects with no control by the older enabled regular realms. So, simulation logs will be created for all mandatory realms. This is the same as the scenario for new mandatory realms with existing enabled mandatory realms.
- New regular realms in simulation mode with existing enabled mandatory and regular realms: The enabled mandatory realms will be the deciding realms whether the new regular realms in simulation mode are added to the existing enabled regular realms in the system. This is the same as the scenario as a mix of mandatory and regular realms, all in simulation mode. Nothing is written to the simulation logs.
- New mandatory realms in simulation mode with enabled mandatory and regular realms: The enabled regular realms can be ignored. This is the same as the scenario for new mandatory realms with existing enabled mandatory realms.
- Mix of new mandatory and regular realms in simulation mode with existing enabled mandatory and regular realms: Ignore all enabled and mandatory regular realms. This is simply adding more mandatory realms to an existing object. This is the same scenario as new mandatory realms with existing enabled mandatory realms.

Use Case: Testing the Addition of New Objects in a Realm

In this use case, you add new objects to an existing realm and then test it using simulation mode without removing the current realm protections.

Oracle recommends that you create a duplicate realm in simulation mode for the new objects with the same authorized users and rule sets. This way, the existing realm can continue to provide protection to the existing objects while testing the new object.

Use Case: Testing the Removal of Objects from a Realm

In this use case, you test the removal of objects to an existing realm.

Because you are removing security controls for an existing object, there is no need to use simulation mode. Simply remove the object from the realm.

Use Case: Testing the Addition of an Authorized User to a Realm

In this use case, you loosen security controls by adding more users. You do not need to simulate anything if you are simply adding more authorized users.

If you are adding new functionality that is accessing data in a realm, but are not sure which new database users to authorize to the realm, then simply run the new functionality as a test (which will be blocked if not already authorized). Review the Database Vault audit logs to see the user name that attempted to access the realm data and add any new database users that are now authorized.

Use Case: Testing the Removal of an Authorized User from a Realm

In this use case, you want to drop an authorized user and use simulation mode to check if the user still needs to access the realm.

You may not be sure about dropping this user because you must check if the authorized user is accessing the realm for authorized activities.

If the data is only protected by a regular realm, then you can clone the realm with authorized users as the only difference. Remove the user to be dropped from the original realm and then add this user to the cloned realm. Then the cloned realm's audit setting is changed to capture `audit on success`. This enables the dropped user to be visible in the audit records if they accessed the realm over a period of time. Audit policies can also be used in this case. For data that is protected by a mandatory realm, the best solution is to create an audit policy.

Use Case: Testing New Factors with Realms

In this use case, you want to test changes to factors.

There are two scenarios where the factors can change:

- Changes to an application or the infrastructure that force a change to the factors

  In this case, you do not need to keep the original factors in place. However, objects and authorized users should be able to remain enabled during the testing of the new factors. With an enabled realm, you can remove the factors from the authorized users. At the same time, create a mandatory realm for the same protected objects in simulation mode with no authorized users. The regular realm will protect the objects from unauthorized users while the simulation realm will capture all access along with the factor information. The simulation log can then be mined for each user to come up with the new factors which can then be added to the mandatory realm in simulation mode to make sure it's clean before being migrated to the original regular realm.

- No changes to the application or the infrastructure but changes such as new factors being added or factors being removed take place

  When factors are being added, you must clone a second simulation realm from the original, but with the new factors added in. If the simulation logs shows that the usage is clean, then you can safely introduce the new factors into the original realm.

  Dropping factors lowers the security profile, so you can simply drop the factor from the rule set. No testing needs to be done.

Use Case: Testing Changes to an Existing Command Rule

In this use case, you test changes to an existing command rule while keeping the original command rule enabled.

Command rules may need to be updated and ideally tested before the changes are enabled in production. For a new command rule that will be added to a set of already existing command rules, put the new command rule into simulation mode when you create it. The other pre-existing command rules are already enabled and offer protection.

If you want to modify an existing command rule, there is no way to maintain the existing protection and test the new modification. Oracle recommends that you create an audit policy to capture what the original command rule was doing and then set an alert for it. The audit will not prevent the SQL as a command rule would do, but at least you can be alerted about the action. Then you can put the new updated command rule into simulation mode and test it.

Tutorial: Tracking Violations to a Realm Using Simulation Mode

This tutorial shows how to create a realm that uses simulation mode and then test violations to the realm.

- About This Tutorial
  In this tutorial, you will create a realm around the `HR.EMPLOYEES` table and test violations against it.
- Step 1: Create Users for This Tutorial
  You must create three users for this tutorial.
- Step 2: Create a Realm and an Oracle Database Vault Policy
  Next, you create a realm around the `HR.EMPLOYEES` table, and then add this realm to an Oracle Database Vault policy.
- Step 3: Test the Realm and Policy
  User `tjones_dba` will commit a violation on the realm to test the realm and policy.
- Step 4: Query the DBA_DV_SIMULATION_LOG View for Violations
  Now you can check the simulation mode log the violations that user `tjones_dba` committed.
- Step 5: Enable and Re-test the Realm
  Now that you have captured the violations, user `psmith` can update the `HR.EMPLOYEES_pol` policy.
- Step 6: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

About This Tutorial

In this tutorial, you will create a realm around the `HR.EMPLOYEES` table and test violations against it.

The `HR.EMPLOYEES` table contains confidential data such employee salaries. To test the realm, an administrator, `tjones_dba`, will look up and modify the salary of another employee, `smavris`. The Database Vault administrator, `leo_dvowner`, will use simulation mode to track the violations to the `HR.EMPLOYEES` table. To accomplish this, user `leo_dvowner` will create a Database Vault policy, which a delegated administrator, user `psmith`, will own. User `psmith` will then be able to make limited changes to the policy without needing the `DV_OWNER` or `DV_ADMIN` role.

Step 1: Create Users for This Tutorial

You must create three users for this tutorial.

The users are: psmith, who is the Database Vault policy owner; tjones_dba, who commits violations on the HR.EMPLOYEES table; and smavris, whose salary is the recipient of tjones_dba's violations.

1. Log into the database instance as a user who has been granted the DV_ACCTMGR role.

   For example:

   ```
   sqlplus bea_dvacctmgr
   Enter password: password
   ```

   In a multitenant environment, you must log in to the appropriate pluggable database (PDB). For example:

   ```
   CONNECT bea_dvacctmgr@hrpdb
   Enter password: password
   ```

   To find the available PDBs, run the `show pdbs` command. To check the current PDB, run the `show con_name` command.

2. Create the following users and grant them the `CREATE SESSION` privilege.

   ```
   GRANT CREATE SESSION TO psmith IDENTIFIED BY password;
   GRANT CREATE SESSION TO tjones_dba IDENTIFIED BY password;
   GRANT CREATE SESSION TO smavris IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

3. Connect as a user who has been granted the DV_OWNER role.

   For example:

   ```
   CONNECT leo_dvowner -- Or, leo_dvowner@hrpdb
   Enter password: password
   ```

4. Grant user psmith the DV_POLICY_OWNER role, which enables psmith to manage Database Vault policies.

   ```
   GRANT DV_POLICY_OWNER TO psmith;
   ```

5. Connect as user SYS with the SYSDBA administrative privilege.

   ```
   CONNECT SYS AS SYSDBA -- Or, CONNECT SYS@hrpdb AS SYSDBA
   Enter password: password
   ```

6. Grant the DBA role to user tjones_dba

   ```
   GRANT DBA TO tjones_dba;
   ```

7. Connect as the HR schema owner.

   ```
   CONNECT HR -- Or, HR@hrpdb
   Enter password: password
   ```

8. Grant the SELECT privilege on the HR.EMPLOYEES table to user smavris

   ```
   GRANT SELECT ON HR.EMPLOYEES TO smavris;
   ```

At this stage, the users have all been created and granted the appropriate privileges.

Step 2: Create a Realm and an Oracle Database Vault Policy

Next, you create a realm around the `HR.EMPLOYEES` table, and then add this realm to an Oracle Database Vault policy.

1. Connect as a user who has been granted the `DV_OWNER` role.

   For example:

   ```
   CONNECT leo_dvowner -- Or, leo_dvowner@hrpdb
   Enter password: password
   ```

2. Create the realm around `HR.EMPLOYEES` table as follows.

   These procedures create the `HR.EMPLOYEES_realm` realm, add the `HR.EMPLOYEES` table to this realm, authenticate `HR` as an owner, authenticate user `psmith` as an participant, and set the realm in simulation mode.

   ```
   BEGIN
    DBMS_MACADM.CREATE_REALM(
     realm_name    => 'HR.EMPLOYEES_realm',
     description   => 'Realm to protect HR.EMPLOYEES',
     enabled       => DBMS_MACUTL.G_SIMULATION,
     audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL,
     realm_type    => 0);
   END;
   /

   BEGIN
    DBMS_MACADM.ADD_OBJECT_TO_REALM(
     realm_name   => 'HR.EMPLOYEES_realm',
     object_owner => 'HR',
     object_name  => 'EMPLOYEES',
     object_type  => 'TABLE');
   END;
   /
   ```

3. Create the `HR.EMPLOYEES_pol` Database Vault policy and set it to be in simulation mode.

   These procedures create the `HR.EMPLOYEES_pol` policy, add the realm that was just created to the policy, and then add user `psmith` as the owner of the policy.

   ```
   BEGIN
    DBMS_MACADM.CREATE_POLICY(
     policy_name  => 'HR.EMPLOYEES_pol',
     description  => 'Policy to protect HR.EMPLOYEES',
     policy_state => DBMS_MACADM.G_SIMULATION);
   END;
   /

   BEGIN
    DBMS_MACADM.ADD_REALM_TO_POLICY(
     policy_name  => 'HR.EMPLOYEES_pol',
     realm_name   => 'HR.EMPLOYEES_realm');
   END;
   /

   BEGIN
   ```

```
 DBMS_MACADM.ADD_OWNER_TO_POLICY(
  policy_name  => 'HR.EMPLOYEES_pol',
  owner_name   => 'PSMITH');
END;
/
```

At this point, the realm and policy are ready to be tested.

Step 3: Test the Realm and Policy

User `tjones_dba` will commit a violation on the realm to test the realm and policy.

1. Connect as user `tjones_dba`.

   ```
   CONNECT tjones_dba -- Or, tjones_dba@hrpdb
   Enter password: password
   ```

2. Query the `HR.EMPLOYEES` table for the salary of `smavris`.

   ```
   SELECT SALARY FROM HR.EMPLOYEES WHERE EMAIL = 'SMAVRIS';
   ```

   Output similar to the following should appear:

   ```
       SALARY
   ----------
         6500
   ```

3. Cut `smavris`'s salary in half.

   ```
   UPDATE HR.EMPLOYEES
   SET SALARY = SALARY / 2
   WHERE EMAIL = 'SMAVRIS';

   1 row updated.
   ```

4. Connect as user `smavris`.

   ```
   CONNECT smavris -- Or, smavris@hrpdb
   ```

5. Query the salary of `smavris`.

   ```
   SELECT SALARY FROM HR.EMPLOYEES WHERE EMAIL = 'SMAVRIS';
   ```

   Output similar to the following should appear:

   ```
       SALARY
   ----------
         3250
   ```

At this point, `tjones_dba`'s violations have been recorded in the `DBA_DV_SIMULATION_LOG` data dictionary view.

Step 4: Query the DBA_DV_SIMULATION_LOG View for Violations

Now you can check the simulation mode log the violations that user `tjones_dba` committed.

1. Connect as a user who has been granted the `DV_OWNER` role.

   For example:

   ```
   CONNECT leo_dvowner -- Or, leo_dvowner@hrpdb
   Enter password: password
   ```

2. Query the DBA_DV_SIMULATION_LOG data dictionary view.

```
SELECT USERNAME, SQLTEXT, VIOLATION_TYPE FROM (
SELECT T1.*, T2.COLUMN_VALUE AS REALM
FROM DVSYS.DBA_DV_SIMULATION_LOG T1, TABLE(T1.REALM_NAME) T2
) WHERE REALM = 'HR.EMPLOYEES_realm';
```

Output similar to the following should appear:

```
USERNAME
--------------------------------------------------------------------------------
SQLTEXT
--------------------------------------------------------------------------------
VIOLATION_TYPE
--------------------------------------------------------------------------------
TJONES_DBA
SELECT SALARY FROM HR.EMPLOYEES WHERE EMAIL = 'SMAVRIS'
Realm Violation

TJONES_DBA
UPDATE HR.EMPLOYEES SET SALARY = SALARY / 2 WHERE EMAIL = 'SMAVRIS'
Realm Violation
```

The output indicates that user tjones_dba has committed two offences: first, he looked at another employee's salary, and not only that, he cut it in half. The violation type is a realm violation. The query by smavris was not captured because she legitimately can look at her salary.

Step 5: Enable and Re-test the Realm

Now that you have captured the violations, user psmith can update the HR.EMPLOYEES_pol policy.

This is so that the HR.EMPLOYEES_realm realm can be enabled. Then you can test the violations again.

1. Connect as user psmith.

```
CONNECT psmith -- Or, psmith@hrpdb
Enter password: password
```

2. Update the policy so that it is enabled.

```
BEGIN
 DBMS_MACADM.UPDATE_POLICY_STATE(
  policy_name   => 'HR.EMPLOYEES_pol',
  policy_state  => 1);
END;
/
```

3. Connect as user tjones_dba.

```
CONNECT tjones_dba --Or, tjones_dba@hrpdb
```

4. Try lowering smavris's salary to new depths.

```
UPDATE HR.EMPLOYEES
SET SALARY = SALARY / 2
WHERE EMAIL = 'SMAVRIS';
```

Output similar to the following should appear:

```
ERROR at line 1:
ORA-01031: insufficient privileges
```

The policy, now enabled, enables the realm to protect the `HR.EMPLOYEES` table. `smavris`'s salary can shrink no more.

Step 6: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. Connect as a user who has been granted the `DV_OWNER` role.

   For example:

   ```
   CONNECT leo_dvowner -- Or, leo_dvowner@hrpdb
   Enter password: password
   ```

2. Remove the `HR.EMPLOYEES_pol` Database Vault policy.

   ```
   EXEC DBMS_MACADM.DROP_POLICY('HR.EMPLOYEES_pol');
   ```

   You first must remove the policy before you can drop its contents.

3. Remove the `HR.EMPLOYEES_realm` realm.

   ```
   EXEC DBMS_MACADM.DELETE_REALM('HR.EMPLOYEES_realm');
   ```

4. Remove the simulation mode log data that was accumulated.

   Because the simulation mode log only captured information about user `tjones_dba`, you can remove only the rows that relate to this user.

   ```
   DELETE FROM DVSYS.SIMULATION_LOG$ WHERE USERNAME = 'TJONES_DBA';
   ```

5. Connect as user `HR`.

   ```
   CONNECT HR -- Or, CONNECT HR@hrpdb
   Enter password: password
   ```

6. Revert `smavris`'s salary back to its pre-violated state.

   ```
   UPDATE HR.EMPLOYEES
   SET SALARY = 6500
   WHERE EMAIL = 'SMAVRIS';
   ```

7. Connect as a user who has been granted the `DV_ACCTMGR` role.

   For example:

   ```
   CONNECT bea_dvacctmgr -- Or, bea_dvacctmgr@hrpdb
   Enter password: password
   ```

8. Remove the users `psmith`, `smavris`, and `tjones_dba`.

   ```
   DROP USER psmith;
   DROP USER smavris;
   DROP USER tjones_dba;
   ```

Use Cases and Guidelines for Logging Realms in Simulation Mode

Oracle provides a set of use cases and guidelines for logging realms in simulation mode.

- About Logging Realms in Simulation Mode
  The type of realm used can affect the data that is written to the simulation log.
- Use Case: New Realms All in Simulation Mode
  In this use case, all realms are either mandatory or regular, and all user-created realms are in simulation mode.
- Use Case: New Realms Introduced to Existing Realms
  The new realms are in simulation mode and the existing realms are enabled and working.
- Use Case: New Objects Added to a Realm
  You can add new objects to an existing realm and then test it by using simulation mode, without removing the current realm protections.
- Use Case: Removing Objects from a Realm
  You can remove objects from a realm by using the `DBMS_MACADM.DELETE_OBJECT_FROM_REALM` procedure.
- Use Case: Adding an Authorized User to a Realm
  You can add authorized users to a realm by using the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure.
- Use Case: Removing an Authorized User from a Realm
  You can remove an authorized user from a realm by using the `DBMS_MACADM.DELETE_AUTH_FROM_REALM` procedure.
- Use Case: Testing New Factors in a Realm
  If a realm has new factors, you can use simulation mode to test if there have been changes to the factors.
- Use Case: Testing Changes to an Existing Command Rule While Keeping the Original Command Control Enabled
  You can test changes to an existing command rule while keeping the original command control enabled.

About Logging Realms in Simulation Mode

The type of realm used can affect the data that is written to the simulation log.

When a user executes a SQL statement and it fails, it may fail for realms that are enabled or fail for realms that are simulated (or both). These could be mandatory realms, regular realms, or both. These conditions determine what is captured in the simulation log.

When an object has both regular realms and mandatory realms protecting it, the mandatory realm protection takes precedence. When both regular and mandatory realms protect an object, the simulation log captures only the activity from the mandatory realm. The simulation log will capture only mandatory realm failures. The only time that the simulation log will capture a regular realm failure is when all realms for the object are regular realms, and if the regular realms have the following behavior:

- All regular realms in simulation mode fail, and
- All regular realms that are enabled also fail

If at least one enabled or simulation regular realm succeeds, then no simulation regular realms are logged in the simulation log.

Use Case: New Realms All in Simulation Mode

In this use case, all realms are either mandatory or regular, and all user-created realms are in simulation mode.

- **Mandatory realms only: all are in simulation mode**
  The user is authorized to execute the SQL statement in all mandatory realms. Nothing is captured in the simulation log table.

  If the user fails one or more mandatory realm checks, then all realm check failures are written to the simulation log. Mandatory realm checks where the user SQL statement succeeded are not written to the simulation log. Suppose there are three mandatory realms, and a user SQL statement succeeds with one realm and fails with the other two. Only the two failed realm checks are written to the simulation log.
- **Regular realms only: all are in simulation mode**
  Suppose the user is authorized to execute the SQL statement in at least one regular realm. This user should have access to the data, so nothing is written the simulation log.

  Alternatively, suppose the user is not authorized to execute the SQL statement of any of the regular realms. The simulation log will capture the failed realm authorization failures. Because the failures are recorded, an administrator can then select which realms to which the user should be authorized. The SQL only needs to be authorized in one regular realm to work. Not all regular realms need to be updated to authorize the SQL.
- **Mix of mandatory and regular realms: all are in simulation mode**
  In this scenario, you capture the gating realms when a user is rejected. With mandatory and regular realms, the mandatory realms are the gating realms. All mandatory realms must pass the authorization check for the user to gain access. In fact, regular realms could be considered superfluous when mandatory realms protect an object. Therefore, in cases where there are mandatory and regular realms, only mandatory realms should be logged into the simulation log . It does not matter whether the user was authorized to the regular realm or not. In this case, only mandatory realms are used and they are all in simulation mode.

  Suppose the user is authorized to execute a SQL statement in all mandatory realms. In this case, nothing is captured in the simulation log table. Even though the user may succeed or fail in one or more regular realms, nothing about regular realm failure is captured.

  If the user fails one or more mandatory realm checks, then all realm check failures are logged to the simulation log. Mandatory realm checks where the user SQL statement succeeded are not logged. For example, suppose there are three mandatory realms. If the user's SQL statement succeeds with one realm and fails with the other two, then only the two failed realm checks are recorded in the simulation log. No regular realms need to be captured, because for this scenario, it does not matter.

Use Case: New Realms Introduced to Existing Realms

The new realms are in simulation mode and the existing realms are enabled and working.

This use case applies only if both simulation mode and enabled realms protect the same object.

- **New enabled mandatory realms with existing enabled mandatory realms** This scenario shows some additional mandatory realms for an existing object, which is designed to add more security to the object.
  - Enabled mandatory realms and mandatory realms in simulation mode are all successful with user's SQL: the SQL executes normally and nothing is written to the simulation log.
  - At least one enabled mandatory realm fails and mandatory realms in simulation mode are successful: the SQL is blocked and nothing is written to the simulation mode log.
  - At least one enabled mandatory realm fails and mandatory realms in simulation mode have one or more failures: the SQL is blocked and all failing mandatory realms in simulation mode are written to the simulation mode log.
  - Enabled mandatory realms are all successful and mandatory realms in simulation mode have at least one failure: the SQL is not blocked, and all failed mandatory realms in simulation mode are written to the simulation log.
- New regular realms in simulation mode with existing enabled regular realms: More regular realms are added to a security object to provide new ways for users to access sensitive data.
  - At least one enabled regular realm and at least one regular realm in simulation mode are successful: the SQL executes normally with nothing is written to the simulation log.
  - At least one enabled regular realm and all regular realms in simulation mode fail: The SQL statement executes normally, so nothing is written to the simulation log.
  - All enabled regular realms and regular realms in simulation mode all fail: the SQL statement is blocked and all regular realms in simulation mode are written to the simulation log. You must evaluate which regular realms to authorize to if necessary. Oracle Database Vault will block the SQL and not record the regular realms in simulation mode in the simulation log because the enabled regular realms would have blocked it anyway. This must change because a user may have added a new realm to authorize the SQL in this use case. There is no way to tell what happened if the new SQL should have worked, but is blocked by all regular realms in simulation mode as well (when one of the regular realms in simulation mode was designed to allow it to work). This would simulate an entry into the audit log for this situation.
  - Enabled regular realms all fail and at least one regular realm in simulation mode successful: the SQL is blocked and nothing is written to the simulation log.
- New regular realm in simulation mode with existing enabled regular realm: This is a do-nothing situation. The enabled regular realm will continue to control the objects and the new regular realm in simulated mode will have no impact if they

are enabled or not. Because there is no impact, no simulation logs are generetaed.

- New mandatory realm in simulated mode with an existing enabled regular realm: While the enabled regular realm will control the objects for now, when the new mandatory realm in simulated mode is enabled, then they will have full control over the objects with no control by the older enabled regular realm. So, simulation logs will be generated for all mandatory realms in simulated mode that fail the authorization checks. This is the same as the use case as with when mandatory realms are all in simulated mode.
- New regular realms in simulated mode with existing enabled mandatory realms in simulated mode and an enabled regular realm: the enabled mandatory realms will be the deciding realms for whether the new simulated mode regular realms are added to the existing enabled regular realms in the system. This is the same as the use case with a new regular realm in simulation mode with existing enabled regular realm. Nothing will be written to the simulation log.
- New mandatory realms in simulated mode with existing enabled mandatory and regular realms: the enabled regular realms can be ignored. This use case is the same as the use case with new simulated mode mandatory realms with existing enabled mandatory realms.
- Mix of new simulated mode mandatory and regular realms with existing enabled mandatory and regular realms: ignore all enabled and simulated mode regular realms. This is simply adding more mandatory realms to an existing object. This use case is the same as the use case with new simulated mandatory realms and enabled mandatory realms.

Use Case: New Objects Added to a Realm

You can add new objects to an existing realm and then test it by using simulation mode, without removing the current realm protections.

Oracle recommends that you create a duplicate realm in simulation mode for the new objects with the same authorized users and rule sets.

Use Case: Removing Objects from a Realm

You can remove objects from a realm by using the `DBMS_MACADM.DELETE_OBJECT_FROM_REALM` procedure.

However, because you are removing security controls for an existing object, there is no need to use simulation mode. All you need to do is remove the object from the realm.

Use Case: Adding an Authorized User to a Realm

You can add authorized users to a realm by using the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure.

In this scenario, you are loosening security controls by adding more users. You do not need to set simulation mode; just add the users. If you do not know which authorized users to add and want to use simulation mode to find out, then Oracle Database Vault enables all users to access the objects. In this case, you must put the original realm in simulation mode.

Use Case: Removing an Authorized User from a Realm

You can remove an authorized user from a realm by using the `DBMS_MACADM.DELETE_AUTH_FROM_REALM` procedure.

In this scenario, you may or may not have dropped a user from a realm authorization list. However, you want to see if the dropped user can still access the realm for authorized activities, as follows:

1. Clone the realm.
2. In the original realm, remove the user that you want to drop and then add this user to the cloned realm.
3. Configure auditing on the cloned realm so that the user that you want to drop is visible in the audit records.
   - If the cloned realm is a regular realm, then use the `DBMS_MACADM.UPDATE_REALM` procedure to set the `audit_options` parameter to `DBMS_MACUTL.G_REALM_AUDIT_SUCCESS`. Alternatively, create a unified audit policy to capture the user's successful activity.
   - If the cloned realm is a mandatory realm, then create a unified audit policy to capture the user's successful activity.

Use Case: Testing New Factors in a Realm

If a realm has new factors, you can use simulation mode to test if there have been changes to the factors.

In this scenario, there are two use cases:

- **There have been changes to the application or infrastructure that will force a change to the factors.**
  In this scenario, enable the (regular) realm and then remove the factors from the authorized users. Then create a mandatory realm for the same protected objects in simulation mode with no authorized users. The regular realm will protect the objects from unauthorized users while the mandatory realm in simulation mode will capture all access, along with any information about the factors. You can check the simulation log for each user to come up with new factors which then can be added to the mandatory realm in simulation mode. This will ensure that the mandatory realm is clean before you migrate it to the original regular realm.
- **There have been no changes to the application or infrastructure, but changes are being made to the factors, such as new factors added or existing factors removed.**
  In this scenario, the concern is about how to configure the addition of new factors. Dropping factors lowers the security profile, so all you need to do is to drop the factors from the rule set. When you add a factor, you must clone a realm from the original and then configure it to be in simulation mode. Add the new factors to this cloned realm. If the simulation log records are clean for this cloned realm, then you can safely add factors to the original realm.

Use Case: Testing Changes to an Existing Command Rule While Keeping the Original Command Control Enabled

You can test changes to an existing command rule while keeping the original command control enabled.

When you update a command rule, ideally you should test it before the changes are enabled and in production. For a new command rule, you only need to put it in simulation mode. However, an existing command rule is already enabled and offering protection. In this case, Oracle recommends that you create an audit policy to capture what the original command rule was doing and set an alert for it. The audit policy will not prevent the SQL from taking place, but at least you will have an alert for it. After you are satisfied that the original command rule works correctly, you can put the updated command rule into simulation mode and then test it.

12

Integrating Oracle Database Vault with Other Oracle Products

You can integrate Oracle Database Vault with other Oracle products, such as Oracle Enterprise User Security.

- Integrating Oracle Database Vault with Enterprise User Security
  You can integrate Oracle Database Vault with Oracle Enterprise User Security.
- Configuring Oracle Database Vault Accounts as Enterprise User Accounts
  You can configure existing Oracle Database Vault user accounts as enterprise user accounts.
- Integration of Oracle Database Vault with Transparent Data Encryption
  Transparent Data Encryption complements Oracle Database Vault in that it provides data protection when the data leaves the secure perimeter of the database.
- Attaching Factors to an Oracle Virtual Private Database
  You can attach factors to an Oracle Virtual Private Database.
- Integrating Oracle Database Vault with Oracle Label Security
  You can integrate Oracle Database Vault with Oracle Label Security, and check the integration with reports and data dictionary views.
- Integrating Oracle Database Vault with Oracle Data Guard
  An Oracle Database Vault-Oracle Data Guard integration requires first, the primary database configuration, then the standby database configration.
- Registering Oracle Internet Directory Using Oracle Database Configuration Asssitant
  You can use Oracle Internet Directory in an Oracle Database Vault-enabled database.

Integrating Oracle Database Vault with Enterprise User Security

You can integrate Oracle Database Vault with Oracle Enterprise User Security.

- About Integrating Oracle Database Vault with Enterprise User Security
  Enterprise User Security centrally manages database users and authorizations in one place.
- Configuring an Enterprise User Authorization
  To configure an Enterprise User authorization, you must create an Oracle Database Vault rule set to control the user access.

About Integrating Oracle Database Vault with Enterprise User Security

Enterprise User Security centrally manages database users and authorizations in one place.

It is combined with Oracle Identity Management and is available in Oracle Database Enterprise Edition.

In general, to integrate Oracle Database Vault with Oracle Enterprise User Security, you configure the appropriate realms to protect the data that you want to protect in the database.

After you define the Oracle Database Vault realms as needed, you can create a rule set for the Enterprise users to allow or disallow their access.

> **See Also:**
>
> *Oracle Database Enterprise User Security Administrator's Guide* for more information about Enterprise User Security

Configuring an Enterprise User Authorization

To configure an Enterprise User authorization, you must create an Oracle Database Vault rule set to control the user access.

1. Create a rule to allow or disallow user access.

   Follow the instructions in Creating a Rule to Add to a Rule Set to create a new rule. In the Create Rule page, enter the following PL/SQL in the Rule Expression field:

   ```
   SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY') = 'user_domain_name'
   ```

   Replace *user_domain_name* with the domain, for example:

   ```
   SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY') = 'myserver.us.example.com'
   ```

2. Add this rule to a new rule set.

   Creating a Rule Set explains how to create a new rule set, including how to add an existing rule to it.

3. Add this rule set to the realm authorization for the data that you want to protect.

   About Realm Authorization explains how to create realm authorizations. In the Authorization Rule Set list, select the rule set that you created in Step 2. Afterward, the realm authorization applies to all users.

Configuring Oracle Database Vault Accounts as Enterprise User Accounts

You can configure existing Oracle Database Vault user accounts as enterprise user accounts.

1. Log into the database instance as a user who has been granted the `CREATE ROLE` system privilege.

   For example:

   ```
   sqlplus system
   Enter password: password
   ```

2. In a multitenant environment, connect to the appropriate pluggable database (PDB).

   For example:

   ```
   CONNECT SYSTEM@hrpdb
   Enter password: password
   ```

   To find the available PDBs, query the `DBA_PDBS` data dictionary view. To check the current PDB, run the `show con_name` command.

3. Create a global role for the `DV_OWNER` role and a global role for the `DV_ACCTMGR` role.

   For example:

   ```
   CREATE ROLE g_dv_owner IDENTIFIED GLOBALLY;
   CREATE ROLE g_dv_acctmgr IDENTIFIED GLOBALLY;
   ```

4. Connect as a user who has been granted the `DV_OWNER` role.

   For example:

   ```
   CONNECT sec_admin_owen -- Or, CONNECT sec_admin_owen@hrpdb
   Enter password: password
   ```

5. Grant the `DV_OWNER` role to the global `DV_OWNER` role.

   ```
   GRANT DV_OWNER TO g_dv_owner;
   ```

6. Connect as a user who has been granted the `DV_ACCTMGR` role.

   For example:

   ```
   CONNECT dbv_acctmgr -- Or, CONNECT dbv_acctmgr@hrpdb
   Enter password: password
   ```

7. Grant the `DV_ACCTMGR` role to the global `DV_ACCTMGR` role.

   ```
   GRANT DV_ACCTMGR TO g_dv_acctmgr;
   ```

8. Connect as user `SYS` with the `SYSDBA` administrative privilege.

   ```
   CONNECT SYS AS SYSDBA -- Or, CONNECT SYS@hrpdb AS SYSDBA
   Enter password: password
   ```

9. Temporarily grant the `DV_ACCTMGR` user who will import the Database Vault users into OID the `CREATE TABLE` privilege and the `SELECT_CATALOG_ROLE` role.

   ```
   GRANT CREATE TABLE, SELECT_CATALOG_ROLE TO dbv_acctmgr;
   ```

10. From the command line, run the User Migration Utility (`UMU`) to import the Database Vault accounts into Oracle Internet Directory (OID).

    The following example imports the Database Vault accounts `leo_dvowner` and `bea_dvacctmgr` into OID. The `DV_ACCTMGR` user is specified for the `DBADMIN` setting.

    ```
    $ORACLE_HOME/rdbms/bin/umu PHASE=ONE
    DBADMIN=dbv_acctmgr:password
    ENTADMIN=cn=jane_ent_admin,dc=example,dc=com:password
    ```

```
USERS= LIST
DBLOCATION=example.com:7777:orcl
DIRLOCATION=example.com:636
USERSLIST=leo_dvowner:bea_dvacctmgr
MAPSCHEMA=PRIVATE
CONTEXT=CONTEXT="c=Users, c=us"
KREALM=EXAMPLE.COM

$ORACLE_HOME/rdbms/bin/umu PHASE=TWO
DBADMIN=dbv_acctmgr:password
ENTADMIN=cn=jane_ent_admin,dc=example,dc=com:password
DBLOCATION=example.com:7777:orcl
DIRLOCATION=example.com:636
```

By default, errors are written to the `$ORACLE_HOME/network/log/umu.log` file.

11. From the Oracle Internet Directory Self Service Console (`http://hostname:port/oiddas/`), grant the global `DV_OWNER` and `DV_ACCTMGR` roles (for example, `g_dv_owner` and `g_dv_acctmgr`) to the enterprise user Database Vault accounts.

    See the example of creating enterprise users in *Oracle Database Enterprise User Security Administrator's Guide* for a demonstration of creating an enterprise role from a global role and then granting this role to a user.

12. From SQL*Plus, as user `SYS` with the `SYSDBA` administrative privilege, revoke the `CREATE TABLE` and `SELECT_CATALOG_ROLE` role from the `DV_ACCTMGR` user.

    ```
    REVOKE CREATE TABLE, SELECT_CATALOG_ROLE FROM dbv_acctmgr;
    ```

> **See Also:**
>
> *Oracle Database Enterprise User Security Administrator's Guide* for detailed information about the User Migration Utility

Integration of Oracle Database Vault with Transparent Data Encryption

Transparent Data Encryption complements Oracle Database Vault in that it provides data protection when the data leaves the secure perimeter of the database.

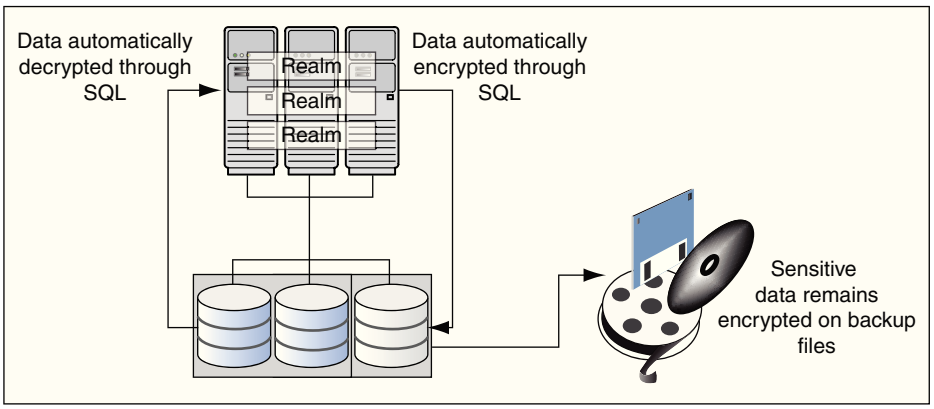
With Transparent Data Encryption, a database administrator or database security administrator can simply encrypt columns with sensitive content in application tables, or encrypt entire application tablespaces, without any modification to the application.

If a user passes the authentication and authorization checks, Transparent Data Encryption automatically encrypts and decrypts information for the user. This way, you can implement encryption without having to change your applications.

Once you have granted the Transparent Data Encryption user the appropriate privileges, then Transparent Data Encryption can be managed as usual and be used complimentary to Database Vault.

Figure 12-1 shows how Oracle Database Vault realms handle encrypted data.

Figure 12-1 Encrypted Data and Oracle Database Vault



> **See Also:**
>
> *Oracle Database Advanced Security Guide* for detailed information about Transparent Data Encryption

Attaching Factors to an Oracle Virtual Private Database

You can attach factors to an Oracle Virtual Private Database.

1. Define a Virtual Private Database policy predicate that is a PL/SQL function or expression.
2. For each function or expression, use the `DVF.F$` PL/SQL function that is created for each factor.

> **See Also:**
>
> *Oracle Database Security Guide* Oracle Database Security Guide for more information about Oracle Virtual Private Database

Integrating Oracle Database Vault with Oracle Label Security

You can integrate Oracle Database Vault with Oracle Label Security, and check the integration with reports and data dictionary views.

- How Oracle Database Vault Is Integrated with Oracle Label Security
  An Oracle Database Vault-Oracle Label Security integration enables you to assign an OLS label to a Database Vault factor identity.

- Requirements for Using Oracle Database Vault with Oracle Label Security
  You must fulfill specific requirements in place before you use Oracle Database Vault with Oracle Label Security.
- Using Oracle Database Vault Factors with Oracle Label Security Policies
  To enhance security, you can integrate Oracle Database Vault factors with Oracle Label Security policies.
- Tutorial: Integrating Oracle Database Vault with Oracle Label Security
  An Oracle Database Vault-Oracle Label Security integration can grant different levels of access to two administrative users who have the same privileges.
- Related Reports and Data Dictionary Views
  Oracle Database Vault provides reports and data dictionary views that list information about the Oracle Database Vault-Oracle Label Security integration.

How Oracle Database Vault Is Integrated with Oracle Label Security

An Oracle Database Vault-Oracle Label Security integration enables you to assign an OLS label to a Database Vault factor identity.

In Oracle Label Security, you can restrict access to records in database tables or PL/SQL programs. For example, Mary may be able to see data protected by the HIGHLY SENSITIVE label, an Oracle Label Security label on the `EMPLOYEE` table that includes records that should have access limited to certain managers. Another label can be PUBLIC, which allows more open access to this data.

In Oracle Database Vault, you can create a factor called Network, for the network on which the database session originates, with the following identities:

- **Intranet:** Used for when an employee is working on site within the intranet for your company.
- **Remote:** Used for when the employee is working at home from a VPN connection.

You then assign a maximum session label to both. For example:

- Assign the Intranet identity to the HIGHLY SENSITIVE Oracle Label Security label.
- Assign the Remote identity to the PUBLIC label.

This means that when Mary is working at home using her VPN connection, she has access only to the limited table data protected under the PUBLIC identity. But when she is in the office, she has access to the HIGHLY SENSITIVE data, because she is using the Intranet identity. Tutorial: Integrating Oracle Database Vault with Oracle Label Security provides an example of how to accomplish this type of integration.

In a non-unified auditing environment, you can audit the integration with Oracle Label Security by using the Label Security Integration Audit Report. Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` table. If unified auditing is enabled, then you can create audit policies to capture this information, as described in *Oracle Database Security Guide*.

See Also:

- Label Security Integration Audit Report
- Oracle Database Vault Oracle Label Security APIs for information about Database Vault APIs that you can use to integrate Database Vault with Oracle Label Security
- Related Reports and Data Dictionary Views for information about reports that you can run on the Oracle Database Vault and Oracle Label Security integration
- *Oracle Label Security Administrator's Guide* for more information about Oracle Label Security labels

Requirements for Using Oracle Database Vault with Oracle Label Security

You must fulfill specific requirements in place before you use Oracle Database Vault with Oracle Label Security.

- Oracle Label Security is licensed separately. Ensure that you have purchased a license to use it.
- Before you install Oracle Database Vault, you must have already installed Oracle Label Security.
- The installation process for Oracle Label Security creates the `LBACSYS` user account. As a user who has been granted the `DV_ACCTMGR` role, unlock this account and grant it a new password. For example:

  ```
  sqlplus bea_dvacctmgr -- Or, sqlplus bea_dvacctmgr@hrpdb for a PDB
  Enter password: password

  ALTER USER LBACSYS ACCOUNT UNLOCK IDENTIFIED BY password;
  ```

  Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.
- If you plan to use the `LBACSYS` user account in Oracle Enterprise Manager, then log into Enterprise Manager as user `SYS` with the `SYSDBA` administrative privilege, and grant this user the `SELECT ANY DICTIONARY` and `SELECT_CATALOG_ROLE` system privileges.
- Ensure that you have the appropriate Oracle Label Security policies defined. For more information, see *Oracle Label Security Administrator's Guide*.
- If you plan to integrate an Oracle Label Security policy with a Database Vault policy, then ensure that the policy name for Oracle Label Security is less than 24 characters. You can check the names of Oracle Label Security policies by querying the `POLICY_NAME` column of the `ALL_SA_POLICIES` data dictionary view.

Using Oracle Database Vault Factors with Oracle Label Security Policies

To enhance security, you can integrate Oracle Database Vault factors with Oracle Label Security policies.

- About Using Oracle Database Vault Factors with Oracle Label Security Policies
  And Oracle Database Vault-Oracle Label Security integration enables you to control the maximum security clearance for a database session.
- Configuring Factors to Work with an Oracle Label Security Policy
  You can define factors that contribute to the maximum allowable data label of an Oracle Label Security policy.

About Using Oracle Database Vault Factors with Oracle Label Security Policies

And Oracle Database Vault-Oracle Label Security integration enables you to control the maximum security clearance for a database session.

Oracle Database Vault controls the maximum security clearance for a database session by merging the maximum allowable data for each label in a database session by merging the labels of Oracle Database Vault factors that are associated to an Oracle Label Security policy.

In brief, a label acts as an identifier for the access privileges of a database table row. A policy is a name associated with the labels, rules, and authorizations that govern access to table rows.

> **See Also:**
>
> *Oracle Label Security Administrator's Guide* for more information about row labels and policies

Configuring Factors to Work with an Oracle Label Security Policy

You can define factors that contribute to the maximum allowable data label of an Oracle Label Security policy.

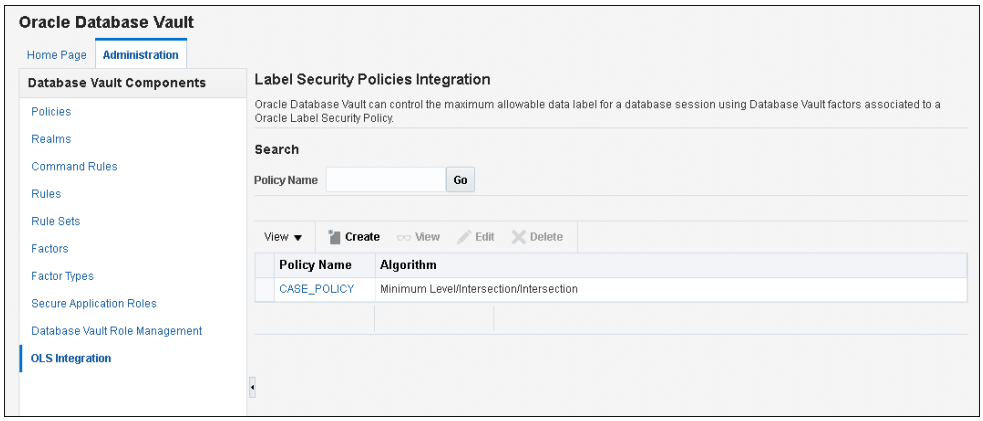
1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. Make the user `LBACSYS` account an owner of the realm that contains the schema to which a label security policy has been applied.

   This enables the `LBACSYS` account to have access to all the protected data in the realm, so that it can properly classify the data.

   The `LBACSYS` account is created in Oracle Label Security using the Oracle Universal Installer custom installation option. Before you can create an Oracle Label Security policy for use with Oracle Database Vault, you must make `LBACSYS`

an owner for the realm you plan to use. See About Realm Authorization for more information.

3. Authorize the schema owner (on which the label security policy has been applied) as either a realm participant or a realm owner.
4. In the Administration page, under Database Vault Components, click **OLS Integration**.



5. In the Label Security Policies Integration page:
   - To register a new label security policy with Database Vault, click **Create**.
   - To edit an existing label security policy that has been registered with Database Vault, select it from the list and then click **Edit**.
6. Enter the following settings:
   - **Label Security Policy:** From the list, select the Oracle Label Security policy that you want to use.
   - **Algorithm:** Optionally change the label-merging algorithm for cases when Oracle Label Security has merged two labels. In most cases, you may want to select **LII - Minimum Level/Intersection/Intersection**. This setting is the most commonly used method that Oracle Label Security administrators use when they want to merge two labels. This setting provides optimum flexibility when your applications must determine the resulting label that is required when combining two data sets that have different labels. It is also necessary for situations in which you must perform queries using joins on rows with different data labels.

     If you want to use the `DBMS_MACADM` package to specify a merge algorithm, see Table 20-2 for a full listing of possible merge algorithms.
   - **Label Security Policy Factors:** In the **Available Factors** list under Label Security Policy Factors, select the factor that you want to associate with the Oracle Label Security policy. Then click **Move** to move the factor to the **Selected Factors** list. You can select multiple factors by holding down the **Ctrl** key as you click each factor that you want to select.
7. Click **OK**.

   The policy is listed in the Label Security Policies Integration page.
8. Label the factor identities using the labels for the policy.

Adding an Identity to a Factor provides detailed information.

> **Note:**
>
> If you do not associate an Oracle Label Security policy with factors, then Oracle Database Vault maintains the default Oracle Label Security behavior for the policy.

Tutorial: Integrating Oracle Database Vault with Oracle Label Security

An Oracle Database Vault-Oracle Label Security integration can grant different levels of access to two administrative users who have the same privileges.

- About This Tutorial
  You can use Oracle Database Vault factors with Oracle Label Security and Oracle Virtual Private Database (VPD) to restrict sensitive data access.
- Step 1: Create Users for This Tutorial
  You must create two administrative users for this tutorial.
- Step 2: Create the Oracle Label Security Policy
  Next, you can create the Oracle Label Security policy and grant users the appropriate privileges for it.
- Step 3: Create Oracle Database Vault Rules to Control the OLS Authorization
  After you create the Oracle Label Security policy, you can create Database Vault rules to work with it.
- Step 4: Update the ALTER SYSTEM Command Rule to Use the Rule Set
  Before the rule set can be used, you must update the ALTER SYSTEM command rule, which is a default command rule.
- Step 5: Test the Authorizations
  With all the components in place, you are ready to test the authorization.
- Step 6: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

About This Tutorial

You can use Oracle Database Vault factors with Oracle Label Security and Oracle Virtual Private Database (VPD) to restrict sensitive data access.

You can restrict this data so that it is only exposed to a database session when the correct combination of factors exists, defined by the security administrator, for any given database session.

Step 1: Create Users for This Tutorial

You must create two administrative users for this tutorial.

1. Log into the database instance as a user who has been granted the `DV_ACCTMGR` role.

   For example:

```
sqlplus bea_dvacctmgr
Enter password: password
```

In a multitenant environment, you must connect to the appropriate pluggable database (PDB).

For example:

```
sqlplus bea_dvacctmgr@hrpdb
Enter password: password
```

To find the available PDBs, query the `DBA_PDBS` data dictionary view. To check the current PDB, run the `show con_name` command.

2. Create the following local users:

```
GRANT CREATE SESSION TO mdale IDENTIFIED BY password CONTAINER = CURRENT;
GRANT CREATE SESSION TO jsmith IDENTIFIED BY password CONTAINER = CURRENT;
```

Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

3. Connect as a user who can grant system privileges and who has been granted the owner authorization for the Oracle System Privilege and Role Management realm, and then grant administrative privileges to users `mdale` and `jsmith`.

```
CONNECT dba_psmith -- Or, CONNECT dba_psmith@hrpdb
Enter password: password

GRANT DBA TO mdale, jsmith;
```

At this stage, users `mdale` and `jsmith` have identical administrative privileges.

Step 2: Create the Oracle Label Security Policy

Next, you can create the Oracle Label Security policy and grant users the appropriate privileges for it.

1. In SQL*Plus, connect as the Oracle Label Security administrator, `LBACSYS`.

```
CONNECT LBACSYS -- Or, CONNECT LBACSYS@hrpdb
Enter password: password
```

If user `LBACSYS` is locked and expired, connect as the Database Vault Account Manager, unlock and unexpire the `LBACSYS` account, and then log back in as `LBACSYS`.

For example:

```
CONNECT bea_dvacctmgr -- Or, CONNECT bea_dvaccmgr@hrpdb
Enter password: password

ALTER USER LBACSYS ACCOUNT UNLOCK IDENTIFIED BY password;

CONNECT LBACSYS
Enter password: password
```

2. Create a new Oracle Label Security policy:

```
EXEC SA_SYSDBA.CREATE_POLICY('PRIVACY','PRIVACY_COLUMN','NO_CONTROL');
```

3. Create the following levels for the `PRIVACY` policy:

```
EXEC SA_COMPONENTS.CREATE_LEVEL('PRIVACY',2000,'S','SENSITIVE');
EXEC SA_COMPONENTS.CREATE_LEVEL('PRIVACY',1000,'C','CONFIDENTIAL');
```

4. Create the PII compartment.

```
EXEC SA_COMPONENTS.CREATE_COMPARTMENT('PRIVACY',100,'PII','PERS_INFO');
```

5. Grant users `mdale` and `jsmith` the following labels:

```
EXEC SA_USER_ADMIN.SET_USER_LABELS('PRIVACY','mdale','S:PII');
EXEC SA_USER_ADMIN.SET_USER_LABELS('PRIVACY','jsmith','C');
```

   User `mdale` is granted the more sensitive label, Sensitive, which includes the PII compartment. User `jsmith` gets the Confidential label, which is less sensitive.

Step 3: Create Oracle Database Vault Rules to Control the OLS Authorization

After you create the Oracle Label Security policy, you can create Database Vault rules to work with it.

1. Connect to SQL*Plus as the Database Vault Owner.

   For example:

```
CONNECT leo_dvowner -- Or, CONNECT leo_dvowner@hrpdb
Enter password: password
```

2. Create the following rule set:

```
EXEC DBMS_MACADM.CREATE_RULE_SET('PII Rule Set', 'Protect PII data from
privileged users','Y',1,0,2,NULL,NULL,0,NULL);
```

3. Create a rule for the PII Rule Set.

```
EXEC DBMS_MACADM.CREATE_RULE('Check OLS Factor',
'dominates(sa_utl.numeric_label(''PRIVACY''),
char_to_label(''PRIVACY'',''S:PII'')) = ''1''');
```

   Ensure that you use single quotes, as shown in this example, and not double quotes.

4. Add the Check OLS Factor rule to the PII Rule Set.

```
EXEC DBMS_MACADM.ADD_RULE_TO_RULE_SET('PII Rule Set', 'Check OLS Factor');
```

Step 4: Update the ALTER SYSTEM Command Rule to Use the Rule Set

Before the rule set can be used, you must update the ALTER SYSTEM command rule, which is a default command rule.

1. As the Database Vault Owner, check the current value of the ALTER SYSTEM command rule, which is one of the default command rules when you install Oracle Database Vault.

```
SELECT * FROM DBA_DV_COMMAND_RULE WHERE COMMAND = 'ALTER SYSTEM';
```

2. Make a note of these settings so that you can revert them to their original values later on.

   In a default installation, the ALTER SYSTEM command rule uses the Allow Fine Grained Control of System Parameters rule set, and is enabled.

3. Update the ALTER SYSTEM command rule to be associated with the PII Rule Set.

```
EXEC DBMS_MACADM.UPDATE_COMMAND_RULE('ALTER SYSTEM', 'PII Rule Set', '%', '%',
'Y');
```

This command adds the PII Rule Set to the ALTER SYSTEM command rule, applies it to all object owners and object names, and enables the command rule.

Step 5: Test the Authorizations

With all the components in place, you are ready to test the authorization.

1. In SQL*Plus, log on as user `mdale`.

   ```
   CONNECT mdale -- Or, CONNECT mdale@hrpdb
   Enter password: password
   ```

2. Check the current setting for the `AUDIT_TRAIL` initialization parameter.

   ```
   SHOW PARAMETER AUDIT_TRAIL

   NAME                                 TYPE        VALUE
   ------------------------------------ ----------- ---------------------
   audit_trail                          string      DB
   ```

   Make a note of this setting, so that you can revert it to its original setting later on.

3. As user `mdale`, use the `ALTER SYSTEM` statement to modify the `CPU_COUNT` parameter.

   ```
   ALTER SYSTEM SET CPU_COUNT = 4;
   System altered.
   ```

   Because user `mdale` was assigned the Sensitive label with the PII compartment, he can use the `ALTER SYSTEM` statement to modify the `AUDIT_TRAIL` system parameter.

4. Set the `CPU_COUNT` parameter back to its original value.

   For example:

   ```
   ALTER SYSTEM SET CPU_COUNT = 2;
   ```

5. Log in as user `jsmith` and then issue the same `ALTER SYSTEM` statement:

   ```
   CONNECT jsmith -- Or, CONNECT jsmith@hrpdb
   Enter password: password

   ALTER SYSTEM SET CPU_COUNT = 14;
   ```

   The following output should appear:

   ```
   ERROR at line 1:
   ORA-01031: insufficient privileges
   ```

   Because user `jsmith` was assigned only the Confidential label, he cannot perform the `ALTER SYSTEM` statement.

Step 6: Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

1. Connect as the Oracle Label Security administrator and remove the label policy and its components.

```
CONNECT LBACSYS -- Or, CONNECT LBACSYS@hrpdb
Enter password: password

EXEC SA_SYSDBA.DROP_POLICY('PRIVACY', TRUE);
```

2. Connect as the Oracle Database Vault Owner and issue the following commands in the order shown, to set the ALTER SYSTEM command rule back to its previous setting and remove the rule set.

   For example:

```
CONNECT leo_dvowner
Enter password: password

EXEC DBMS_MACADM.UPDATE_COMMAND_RULE('ALTER SYSTEM', 'Allow System
Parameters','%', '%', 'Y');
EXEC DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('PII Rule Set', 'Check OLS Factor');
EXEC DBMS_MACADM.DELETE_RULE('Check OLS Factor');
EXEC DBMS_MACADM.DELETE_RULE_SET('PII Rule Set');
COMMIT;
```

3. Connect as the Database Vault Account Manager and remove users `mdale` and `jsmith`.

```
CONNECT bea_dvacctmgr -- Or, CONNECT bea_dvacctmgr@hrpdb
Enter password: password

DROP USER mdale;
DROP USER jsmith;
```

Related Reports and Data Dictionary Views

Oracle Database Vault provides reports and data dictionary views that list information about the Oracle Database Vault-Oracle Label Security integration.

Table 12-1 lists the Oracle Database Vault reports. See Oracle Database Vault Reports , for information about how to run these reports.

Table 12-1 Reports Related to Database Vault and Oracle Label Security Integration

Report	Description
Factor Configuration Issues Report	Lists factors in which the Oracle Label Security policy does not exist.
Identity Configuration Issues Report	Lists invalid label identities (the Oracle Label Security label for this identity has been removed and no longer exists).
Security Policy Exemption Report	Lists accounts and roles that have the `EXEMPT ACCESS POLICY` system privilege granted to them. Accounts that have this privilege can bypass all Virtual Private Database policy filters and any Oracle Label Security policies that use Oracle Virtual Private Database indirectly.

Table 12-2 lists data dictionary views that provide information about existing Oracle Label Security policies used with Oracle Database Vault.

Table 12-2 Data Dictionary Views Used for Oracle Label Security

Data Dictionary View	Description
DBA_DV_MAC_POLICY View	Lists the Oracle Label Security policies defined
DBA_DV_MAC_POLICY_FACTOR View	Lists the factors that are associated with Oracle Label Security policies
DBA_DV_POLICY_LABEL View	Lists the Oracle Label Security label for each factor identifier in the `DBA_DV_IDENTITY` view for each policy

Integrating Oracle Database Vault with Oracle Data Guard

An Oracle Database Vault-Oracle Data Guard integration requires first, the primary database configuration, then the standby database configration.

- Step 1: Configure the Primary Database
  You must run the DGMGRL and DBCA utilities, and then the `ALTER SYSTEM` statement, to configure the primary database.
- Step 2: Configure the Standby Database
  You can perform the standby database configuration within the database to be used for the standby database.

Step 1: Configure the Primary Database

You must run the DGMGRL and DBCA utilities, and then the `ALTER SYSTEM` statement, to configure the primary database.

1. For Linux and UNIX systems, ensure there is an `/etc/oratab` entry for the database on the node in which you are installing Oracle Database Vault.
2. If you are using Data Guard Broker, then from the command prompt, disable the configuration as follows:

   ```
   dgmgrl sys
   Enter password: password

   DGMGRL> disable configuration;
   ```

3. Run Database Configuration Assistant (DBCA) and configure the database options to add Oracle Database Vault to the primary database.
   a. From the command line, enter the following command to start DBCA:

      ```
      dbca
      ```

   b. Select the correct database type (**Cluster** or **Single Instance**) and click **Next**.
   c. In the Database Operation page, select **Configure Database Options** and click **Next**.
   d. Select the appropriate database and click **Next**.
   e. Select **Oracle Label Security**, which then enables you to select **Oracle Database Vault** and click **Next**.

f. Enter the name of the Database Vault owner (required) and the Database Vault account manager (recommended).

   Passwords must have at least one alphabetic character, one number, and one special character.

g. Click **Next**.

h. Choose appropriate connection mode and click **Next**.

i. Click **OK** to restart the database.

j. Click **OK** on **Configure Additional Components**.

At this point, the installation on the primary site is complete.

4. Log into the database instance as user `SYS` with the `SYSDBA` administrative privilege.

```
sqlplus sys as sysdba
Enter password: password
```

5. Run the following `ALTER SYSTEM` statements:

```
ALTER SYSTEM SET AUDIT_SYS_OPERATIONS=TRUE SCOPE=SPFILE;
ALTER SYSTEM SET OS_ROLES=FALSE SCOPE=SPFILE;
ALTER SYSTEM SET RECYCLEBIN='OFF' SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_LOGIN_PASSWORDFILE='EXCLUSIVE' SCOPE=SPFILE;
ALTER SYSTEM SET SQL92_SECURITY=TRUE SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_OS_AUTHENT=FALSE SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_OS_ROLES=FALSE SCOPE=SPFILE;
```

6. Run the `ALTER SYSTEM` statement on each database instance to set the parameters as shown in Step 5.

7. Restart each database instance.

```
CONNECT SYS AS SYSOPER
Enter password: password

SHUTDOWN IMMEDIATE
STARTUP
```

Step 2: Configure the Standby Database

You can perform the standby database configuration within the database to be used for the standby database.

1. Log into the database instance as user `SYS` with the `SYSDBA` administrative privilege.

```
sqlplus sys as sysdba
Enter password: password
```

2. In a multitenant environment, connect to the appropriate PDB.

   For example:

```
CONNECT bea_dvacctmgr@hrpdb
Enter password: password
```

   To find the available PDBs, run the `show pdbs` command. To check the current PDB, run the `show con_name` command.

3. Mount a standby database instance.

```
ALTER DATABASE MOUNT STANDBY DATABASE;
```

4. Run the following `ALTER SYSTEM` statements:

```
ALTER SYSTEM SET AUDIT_SYS_OPERATIONS=TRUE SCOPE=SPFILE;
ALTER SYSTEM SET OS_ROLES=FALSE SCOPE=SPFILE;
ALTER SYSTEM SET RECYCLEBIN='OFF' SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_LOGIN_PASSWORDFILE='EXCLUSIVE' SCOPE=SPFILE;
ALTER SYSTEM SET SQL92_SECURITY=TRUE SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_OS_AUTHENT=FALSE SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_OS_ROLES=FALSE SCOPE=SPFILE;
```

5. Restart or mount the database instance.

   For example:

```
SHUTDOWN IMMEDIATE
STARTUP
```

6. Mount the next standby instance.
7. Restart the managed recovery as follows:

```
ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
```

8. If you are using Data Guard Broker, then from the command line, re-enable the configuration.

```
dgmgrl sys
Enter password: password

DGMGRL> enable configuration;
```

   This command applies the changes to the physical standby database made by the Oracle Database Vault installation on the primary database.

9. Repeat the physical standby installation process on each physical standby database. For example, if there are three physical standby databases, then run these procedures or each standby database.

Registering Oracle Internet Directory Using Oracle Database Configuration Asssitant

You can use Oracle Internet Directory in an Oracle Database Vault-enabled database.

However, if you want to register Oracle Internet Directory (OID) using Oracle Database Configuration Assistant (DBCA), then you must first disable Oracle Database Vault.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13

DBA Operations in an Oracle Database Vault Environment

Database administrators can perform operations in an Oracle Database Vault environment, such as using Database Vault with products such as Oracle Data Pump.

- Using Oracle Database Vault with Oracle Enterprise Manager
  Oracle Database Vault administrators can perform tasks in Oracle Enterprise Manager Cloud Control such as propagating polices to other databases.
- Using Oracle Data Pump with Oracle Database Vault
  Database administrators can authorize Oracle Data Pump users to work in a Database Vault environment.
- Using Oracle Scheduler with Oracle Database Vault
  Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific authorization.
- Using Information Lifecycle Management with Oracle Database Vault
  Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.
- Using Oracle Database Replay with Oracle Database Vault
  Database administrators can authorize Oracle Database Replay users to work in a Database Vault environment.
- Executing Preprocessor Programs with Oracle Database Vault
  Users who execute preprocessor programs through external tables must have Oracle Database Vault-specific authorization.
- Oracle Recovery Manager and Oracle Database Vault
  You can use Recovery Manager (RMAN) in an Oracle Database Vault environment.
- Privileges for Using Oracle Streams with Oracle Database Vault
  If you want to use Oracle Streams in an Oracle Database Vault environment, then you must have the correct privileges.
- Privileges for Using XStream with Oracle Database Vault
  If you want to use XStream in an Oracle Database Vault environment, then you must have the appropriate privileges.
- Privileges for Using Oracle GoldenGate in with Oracle Database Vault
  If you want to use Oracle GoldenGate in an Oracle Database Vault environment, then you must have the appropriate privileges.
- Using Data Masking in an Oracle Database Vault Environment
  You must have the correct authorization to perform data masking in an Oracle Database Vault environment.
- Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB
  You can convert a standalone Oracle Database Release 12c or later database to a PDB, and then plug this PDB into a CDB.

- Using the ORADEBUG Utility with Oracle Database Vault
  The `ORADEBUG` utility is used primarily by Oracle Support to diagnose problems that may arise with an Oracle database.

Using Oracle Database Vault with Oracle Enterprise Manager

Oracle Database Vault administrators can perform tasks in Oracle Enterprise Manager Cloud Control such as propagating polices to other databases.

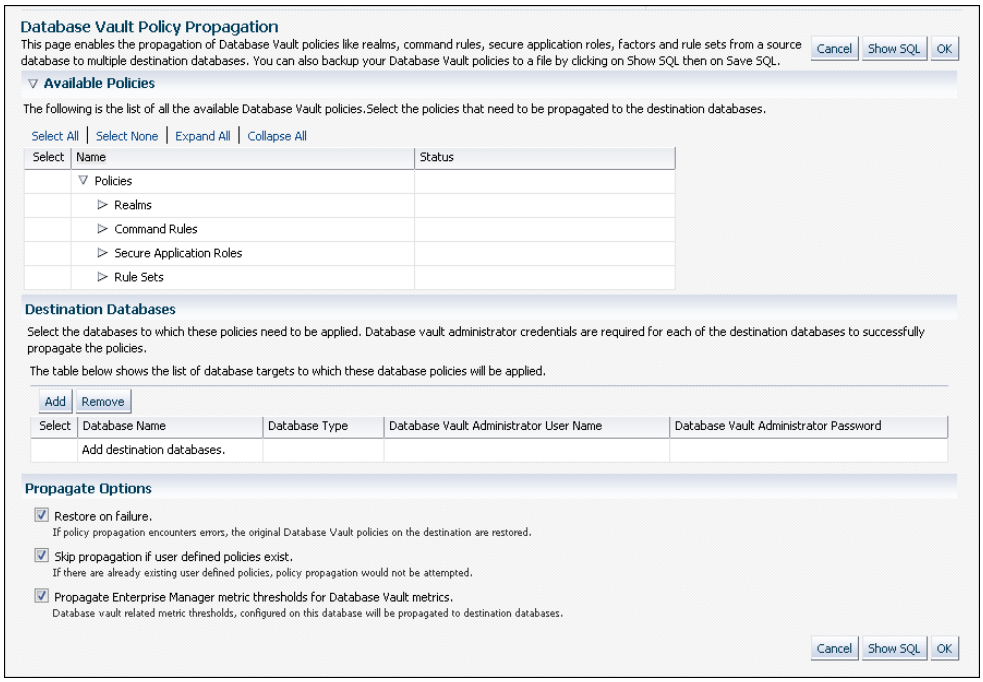
- Propagating Oracle Database Vault Configurations to Other Databases
  You can propagate Database Vault configurations (such as a realm configuration) to other Database Vault-protected databases.
- Enterprise Manager Cloud Control Alerts for Oracle Database Vault Policies
  To view Oracle Database Vault alerts, you must be granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.
- Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control
  From the Database Vault home page, you can find information about violations.
- Changing the DBSNMP Account Password in a Database Vault Environment
  Before you can change the password for the `DBSNMP` user account, you must revoke the `DV_MONITOR` role from this account.

Propagating Oracle Database Vault Configurations to Other Databases

You can propagate Database Vault configurations (such as a realm configuration) to other Database Vault-protected databases.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Database Vault home page, under Database Vault Policy Propagation, select **Database Vault Policy Propagation**.

   The Available Policies area in the Policy Propagation subpage lists a summary of the Oracle Database Vault configurations that were created for the current database: that is, configurations that were created for realms, command rules, rule sets, and secure application roles. It does not list the Oracle Database Vault policies that were introduced in Oracle Database release 12*c* (12.2). From here, you can propagate these configurations to another database.
3. Under Available Policies, select each configuration that you want to propagate to another database.



4. Under Destination Databases, click the **Add** button.
5. Under Search and Select: Database Vault Enabled Destination Databases, search for the destination databases, and then select each database to which you want to propagate the configurations. Then click the **Select** button.
6. Under Destination Databases, do the following:
   a. Under Apply credentials across destination database(s), enter the user name and password of the administrator of the Database Vault database that contains the configurations you want to propagate.

      This feature applies the Database Vault administrator's user name and password to all of the selected destination databases.

   b. Select each database to which you want to propagate the configurations.
   c. Enter the Database Vault administrator user name and password for each database.
   d. Click the **Apply** button.
7. In the Propagate Options page, select from the following options.

   Any changes made to the seeded realms, command rules, rule sets, and so on will not be propagated to the destination databases. Only custom-created data are propagated.

   - **Restore on failure:** If the propagation operations encounters errors, then the propagation is rolled back. That is, the original policies on the destination database are restored. If you do not select this option, then the policy propagation on the destination database continues and ignores any errors.
   - **Skip propagation if user defined policies exist:** If the destination databases already have the user-defined configurations, then the propagation operation is not attempted. If you do not select this option, then regardless of whether user-defined policies exist on the destination database, all the existing

configurations are cleared, and the configurations from the source database are applied to the destination database.

- **Propagate Enterprise Manager metric thresholds for database vault metrics:** If the source database has Oracle Database Vault metric thresholds set, then these thresholds are also propagated to the destination databases. If you do not select this option, then only configurations are propagated and not the Oracle Database Vault thresholds.

8. Click the **OK** button.
9. In the Confirmation window, click **OK**.

   A message indicating success or failure appears. If the propagation succeeds, then the configurations are active right away in their destination databases.

Enterprise Manager Cloud Control Alerts for Oracle Database Vault Policies

To view Oracle Database Vault alerts, you must be granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

The alerts are as follows:

- **Database Vault Attempted Realm Violations.** This alert helps the Oracle Database Vault security analyst (`DV_SECANALYST` role) to monitor violation attempts on the Database Vault database. This user can select the realms to be affected by the alert and filter these realms based on the different types of attempts by using error codes. You can enable this metric from the Metrics and Policy Settings page. By default, the attempted realm violations are collected every 24 hours.
- **Database Vault Attempted Command Rule Violations.** The functionality for this alert is the same as for Database Vault Attempted Realm Violations, except that it focuses on violations on command rules.
- **Database Vault Realm Configuration Issues.** This metric tracks and raises an alert if users misconfigure realms. This metric is enabled when you install Oracle Database vault, and by default it collects data every one hour.
- **Database Vault Command Rule Configuration Issues.** This functionality for this alert is that same as Database Vault Realm Configuration Issues, except that it focuses on configuration changes to command rules.
- **Database Vault Policy Changes.** This metric raises an alert on any change to any Database Vault policy, such as policies for realms and command rules. It provides a detailed policy changes report.

Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control

From the Database Vault home page, you can find information about violations.

These violations are as follows:

- Top five attempted violations on realm and command rule
- Top five attempted violations by database users and client host

- Time series-based graphical reports on attempted violations for more detailed analysis

To have full access to the Database Vault reports, you must log into Database Vault Administrator as a user who has been granted the DV_OWNER, DV_ADMIN, or DV_SECANALYST role.

Related Topics

- Oracle Database Vault Reports
  Oracle Database Vault provides reports that track activities, such as the Database Vault configuration settings.

Changing the DBSNMP Account Password in a Database Vault Environment

Before you can change the password for the DBSNMP user account, you must revoke the DV_MONITOR role from this account.

In an Oracle Database Vault environment, the DBSNMP user account is granted the DV_MONITOR role. (The DBSNMP user can change his or her own password directly, without having to have the DV_MONITOR role revoked first.)

1. Log into the database instance using an account that has been granted the DV_OWNER role.
2. Revoke the DV_MONITOR role from the DBSNMP user account.
3. Connect as a user who has been granted the DV_ACCTMGR role and then change the DBSNMP user account password.
4. Connect as the DV_OWNER user and then grant the DV_MONITOR role back to the DBSNMP user account.

Using Oracle Data Pump with Oracle Database Vault

Database administrators can authorize Oracle Data Pump users to work in a Database Vault environment.

- About Using Oracle Data Pump with Oracle Database Vault
  Database administrators who use Oracle Data Pump in an Database Vault environment must have Database Vault-specific authorization to export and import data.
- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.
- Authorizing Users or Roles for Data Pump Transportable Export and Import Operations
  You can grant authorization levels for users who must perform Oracle Data Pump transportable operations, either directly or through a role.
- Guidelines for Exporting or Importing Data in a Database Vault Environment
  After you grant the Oracle Data Pump database administrator the proper authorization, this user can perform any export or import operations that are necessary.

About Using Oracle Data Pump with Oracle Database Vault

Database administrators who use Oracle Data Pump in an Database Vault environment must have Database Vault-specific authorization to export and import data.

This type of user must have Database Vault privileges in addition to the standard Oracle Data Pump privileges. If these users want to perform Oracle Data Pump transportable tablespace operations, then they must have special authorization. You can check a user's authorizations for using Data Pump in an Oracle Database Vault environment by querying the `DBA_DV_DATAPUMP_AUTH` data dictionary view. You can grant this authorization to either individual users or to database roles.

See Also:

- *Oracle Database Utilities* for detailed information about Oracle Data Pump
- *Oracle Database Administrator's Guide* for more information about transportable tablespaces
- DBA_DV_DATAPUMP_AUTH View

Authorizing Users or Roles for Data Pump Regular Export and Import Operations

You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

- About Authorizing Users or Roles for Oracle Data Pump Regular Operations
  Users who have Oracle Data Pump authorization can perform regular Oracle Data Pump operations in a Database Vault environment.
- Levels of Database Vault Authorization for Oracle Data Pump Regular Operations
  Oracle Database Vault provides several levels of authorization required for Oracle Data Pump regular operations in a Database Vault environment.
- Authorizing Users or Roles for Oracle Data Pump Regular Operations in Database Vault
  You can authorize a database administrator or a role to use Data Pump for regular operations in an Oracle Database Vault environment.
- Revoking Oracle Data Pump Authorization from Users or Roles
  You can revoke authorization from the database administrator or role who is using Oracle Data Pump for regular operations.

About Authorizing Users or Roles for Oracle Data Pump Regular Operations

Users who have Oracle Data Pump authorization can perform regular Oracle Data Pump operations in a Database Vault environment.

Full level Data Pump authorization enables these users to perform transportable export and import operations as well.

Related Topics

- Authorizing Users or Roles for Data Pump Transportable Export and Import Operations
  You can grant authorization levels for users who must perform Oracle Data Pump transportable operations, either directly or through a role.

Levels of Database Vault Authorization for Oracle Data Pump Regular Operations

Oracle Database Vault provides several levels of authorization required for Oracle Data Pump regular operations in a Database Vault environment.

Table 13-1 describes these levels.

Table 13-1 Levels of Authorization for Oracle Data Pump Regular Operations

Scenario	Authorization Required
A database administrator wants to import data into another schema.	You must grant this user (or a role) the `BECOME USER` system privilege and the `IMP_FULL_DATABASE` role.[1] To find the privileges a user has been granted, query the `USER_SYS_PRIVS` data dictionary view.
A database administrator wants to export or import data in a schema that has no Database Vault protection.	You only need to grant this user (or a role) the standard Oracle Data Pump privileges, which are the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles. If the user wants to import data, grant this user the `BECOME USER` system privilege.
A database administrator wants to export or import data in a protected schema.	In addition to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles, you must grant this user (or a role) Database Vault-specific authorization by using the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure. This authorization applies to both the `EXPDP` and `IMPDP` utilities. Later on, you can revoke this authorization by using the `DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER` procedure. If the user wants to import data, also grant this user the `BECOME USER` system privilege.
A database administrator wants to export or import the contents of an entire database.	In addition to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles and the authorization granted by the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure, you must grant this user (or a role) the `DV_OWNER` role. If the user wants to import data, grant this user the `BECOME USER` system privilege.

[1] The `BECOME USER` privilege is part of the `IMP_FULL_DATABASE` role by default, but in an Oracle Database Vault environment, this privilege is revoked.

Authorizing Users or Roles for Oracle Data Pump Regular Operations in Database Vault

You can authorize a database administrator or a role to use Data Pump for regular operations in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. Ensure that the user or role to whom you want to grant authorization has been granted the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles, which are required for using Oracle Data Pump.

   ```
   SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE LIKE '%FULL
   %';
   ```

3. Grant this user or role Oracle Database Vault authorization for Oracle Data Pump regular operations.

   For example, to authorize the Data Pump user DP_MGR to export and import objects for the database table EMPLOYEES:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR', 'EMPLOYEES');
   ```

   To restrict DP_MGR's activities to a specific schema, you would enter the following procedure:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR');
   ```

   To authorize users who have been granted the DP_MGR_ROLE role to export and import objects for the entire database, enter the following:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR_ROLE');
   ```

   After you run the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure, you can check the authorization of the user or role by querying the DBA_DV_DATAPUMP_AUTH data dictionary view.
4. If the user must export the entire database, then grant the user (or role) the DV_OWNER role.

   For example, for a role:

   ```
   GRANT DV_OWNER TO DP_MGR_ROLE;
   ```

Related Topics

- AUTHORIZE_DATAPUMP_USER Procedure
  The AUTHORIZE_DATAPUMP_USER procedure authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled.
- DBA_DV_DATAPUMP_AUTH View
  The DBA_DV_DATAPUMP_AUTH data dictionary view lists the authorizations for using Oracle Data Pump in an Oracle Database Vault environment.

Revoking Oracle Data Pump Authorization from Users or Roles

You can revoke authorization from the database administrator or role who is using Oracle Data Pump for regular operations.

1. If you granted the user or role the DV_OWNER role, then optionally revoke the DV_OWNER role.

   ```
   REVOKE DV_OWNER FROM DP_MGR;
   ```

2. Query the DBA_DV_DATAPUMP_AUTH data dictionary view to find the users or roles that have been granted Oracle Data Pump authorizations.

   ```
   SELECT GRANTEE, SCHEMA, OBJECT FROM DBA_DV_DATAPUMP_AUTH;
   ```

3. Use the information you gathered from the preceding step to build the DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER command.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR', 'EMPLOYEES');
   ```

   Ensure that this unauthorization complements the original authorization action. In other words, if you originally gave DP_MGR authorization over the entire database, then the following commands will not work:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR');

   EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR', 'EMPLOYEES');
   ```

Related Topics

- UNAUTHORIZE_DATAPUMP_USER Procedure
  The UNAUTHORIZE_DATAPUMP_USER procedure revokes the authorization that was granted by the AUTHORIZE_DATAPUMP_USER procedure.
- DBA_DV_DATAPUMP_AUTH View
  The DBA_DV_DATAPUMP_AUTH data dictionary view lists the authorizations for using Oracle Data Pump in an Oracle Database Vault environment.

Authorizing Users or Roles for Data Pump Transportable Export and Import Operations

You can grant authorization levels for users who must perform Oracle Data Pump transportable operations, either directly or through a role.

- About Authorizing Users for Oracle Data Pump Transportable Operations
  You can grant users (either directly or through a role) different levels of transportable operation authorization.
- Levels of Database Vault Authorization for Data Pump Transportable Operations
  Oracle Database Vault provides levels of authorization required for users who must perform export and import transportable operations in a Database Vault environment.
- Authorizing Users or Roles for Data Pump Transportable Operations in Database Vault
  You can authorize users or roles to perform Oracle Data Pump transportable export or import operations in a Database Vault environment.
- Revoking Transportable Tablespace Authorization from Users or Roles
  You can revoke authorization from the database administrator who is using Data Pump.

About Authorizing Users for Oracle Data Pump Transportable Operations

You can grant users (either directly or through a role) different levels of transportable operation authorization.

If you want users to only have the authorization to perform transportable export and import operations, then you must grant users or roles the correct authorization, based on their tasks.

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

Levels of Database Vault Authorization for Data Pump Transportable Operations

Oracle Database Vault provides levels of authorization required for users who must perform export and import transportable operations in a Database Vault environment.

Table 13-2 describes these levels.

Table 13-2 Levels of Authorization for Oracle Data Pump Transporatable Operations

Scenario	Authorization Required
A database administrator wants to transportable export a tablespace or table that has no Database Vault protection.	You only need to grant this user (or a role) the standard Oracle Data Pump privileges, which are the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles.
A database administrator wants to transportable export a tablespace where there is Database Vault protection (for example, realm or command rule for a table object residing on that tablespace).	In addition to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles, you must grant this user (or a role) Database Vault-specific transportable tablespace authorization by using the `DBMS_MACADM.AUTHORIZE_TTS_USER` procedure. Later on, you can revoke this authorization by using the `DBMS_MACADM.UNAUTHORIZE_TTS_USER` procedure. Remember that users who have been granted full database level Oracle Data Pump authorization (through the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure) can perform these operations as well.
A database administrator wants to transportable export a table within a tablespace where there is Database Vault protection (for example, a realm or command rule for a table object residing on the tablespace that contains the table to be exported).	In addition to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles, you must grant this user (or a role) Database Vault-specific transportable tablespace authorization for the tablespace that contains the table to be exported by using the `DBMS_MACADM.AUTHORIZE_TTS_USER` procedure. Remember that users who have been granted full database level Oracle Data Pump authorization (from the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure) can perform these operations as well.

Table 13-2 (Cont.) Levels of Authorization for Oracle Data Pump Transporatable Operations

Scenario	Authorization Required
A database administrator wants to transportable export the contents of an entire database.	In addition to the DV_OWNER, EXP_FULL_DATABASE, and IMP_FULL_DATABASE roles, you must grant this user (or a role) Database Vault-specific full database level Oracle Data Pump authorization by using the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure. You do not need to run the DBMS_MACADM.AUTHORIZE_TTS_USER procedure for this user.
A database administrator wants to use a network link to transportable import a tablespace or a table that has no Database Vault protection.	In addition to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles for both the database administrator and the connecting user, you must grant the connecting user (or a role) specified in the network link the DV_DATAPUMP_NETWORK_LINK role.
A database administrator wants to use a network link to transportable import a tablespace where there is Database Vault protection (for example, realm or command rule for a table object residing on that tablespace)	In addition to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles, you must grant the connecting user (or a role) specified in the network link the Database Vault-specific transportable tablespace authorization for that tablespace by using the DBMS_MACADM.AUTHORIZE_TTS_USER procedure. You must also grant the connecting user the DV_DATAPUMP_NETWORK_LINK role. Remember that users that have been granted Database Vault-specific full database level Oracle Data Pump authorization (through the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure) can perform these operations.
A database administrator wants to use a network link to import a table within a transportable tablespace where there is Database Vault protection (for example, realm or command rule for a table object residing on the tablespace that contains the table to be exported)	In addition to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles, you must grant the connecting user (or a role) the Database Vault-specific transportable tablespace authorization for the tablespace that contains the table to be exported by using the DBMS_MACADM.AUTHORIZE_TTS_USER procedure. You also must grant the connecting user (or a role) specified in the network link the DV_DATAPUMP_NETWORK_LINK role. Remember that users who have been granted Database Vault-specific full database level Oracle Data Pump authorization (through the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure) can perform the operations.
A database administrator wants to use a network link to transportable import the contents of an entire database.	In addition to the DV_OWNER role, you must grant the connecting user (or a role) Database Vault-specific full database level Oracle Data Pump authorization by using the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure. You do not need to run the DBMS_MACADM.AUTHORIZE_TTS_USER procedure for this user. You must also grant the connecting user (or a role) who is specified in the network link the DV_DATAPUMP_NETWORK_LINK role.

Authorizing Users or Roles for Data Pump Transportable Operations in Database Vault

You can authorize users or roles to perform Oracle Data Pump transportable export or import operations in a Database Vault environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

2. Ensure that the user or role to whom you want to grant authorization has been granted the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles, which are required for using Oracle Data Pump.

```
SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS
 WHERE GRANTED_ROLE LIKE '%FULL%';
```

3. If the user wants to transportable export or use a network link to transportable import the contents of an entire database, then grant the full database level Oracle Data Pump authorization to the user or role by using the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure. Otherwise, bypass this step.

   For example:

```
EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR');
```

4. If the user must have Database Vault-specific transportable tablespace authorization only, then grant this user or role this authorization.

   For example:

```
EXEC DBMS_MACADM.AUTHORIZE_TTS_USER('DP_MGR', 'HR_TS');
```

5. If the user who wants to perform a transportable import operation wants to use a network link to perform the operation, then grant this user or role the DV_DATAPUMP_NETWORK_LINK role.

   For example:

```
GRANT DV_DATAPUMP_NETWORK_LINK TO DP_MGR;
```

6. If the user wants to perform a transportable export or use a network link to transportable import the entire database, then grant this user or role the DV_OWNER role.

```
GRANT DV_OWNER TO DP_MGR;
```

Related Topics

- AUTHORIZE_TTS_USER Procedure
  The AUTHORIZE_TTS_USER procedure authorizes a user to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled.
- AUTHORIZE_DATAPUMP_USER Procedure
  The AUTHORIZE_DATAPUMP_USER procedure authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled.
- DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role
  The DV_DATAPUMP_NETWORK_LINK role is used for Data Pump import operations.

Revoking Transportable Tablespace Authorization from Users or Roles

You can revoke authorization from the database administrator who is using Data Pump.

1. If you granted the user or role the DV_OWNER role, then optionally revoke this role.

```
REVOKE DV_OWNER FROM DP_MGR;
```

2. Query the DBA_DV_TTS_AUTH data dictionary view to find the users and roles that have been granted Oracle Data Pump authorizations.

```
SELECT GRANTEE, TSNAME FROM DBA_DV_TTS_AUTH;
```

3. Use the information you gathered from the preceding step to build the `DBMS_MACADM.UNAUTHORIZE_TTS_USER` statement.

   For example:

   ```
   EXEC DBMS_MACADM.UNUTHORIZE_TTS_USER('DP_MGR', 'HR_TS');
   ```

4. If the user had transportable exported or used a network link to transportable import the contents of an entire database, then revoke the full database level Oracle Data Pump authorization from the user or role.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('DP_MGR');
   ```

5. If the user who had performed a transportable import operation used a network link to perform the operation, then revoke the `DV_DATAPUMP_NETWORK_LINK` role from the user or role.

   For example:

   ```
   REVOKE DV_DATAPUMP_NETWORK_LINK FROM DP_MGR;
   ```

Related Topics

- UNAUTHORIZE_TTS_USER Procedure
  The `UNAUTHORIZE_TTS_USER` procedure removes from authorization users who had previously been granted the authorization to perform Oracle Data Pump transportable tablespace operations.
- UNAUTHORIZE_DATAPUMP_USER Procedure
  The `UNAUTHORIZE_DATAPUMP_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_DATAPUMP_USER` procedure.
- DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role
  The `DV_DATAPUMP_NETWORK_LINK` role is used for Data Pump import operations.

Guidelines for Exporting or Importing Data in a Database Vault Environment

After you grant the Oracle Data Pump database administrator the proper authorization, this user can perform any export or import operations that are necessary.

Before this user begins work, he or she should follow these guidelines:

- **Create a full backup of the database datafiles.** This way, if you or other users do not like the newly-imported data, then you easily can revert the database to its previous state. This guideline is especially useful if an intruder had managed to modify Oracle Data Pump exported data to use his or her own policies.
- **Decide how to handle exporting and importing multiple schemas or tables.** You cannot specify multiple schemas or tables in the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure, but you can use either of the following methods to accomplish this task:
  - Run the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure for each schema or table, and then specify the list of these objects in the `SCHEMAS` or `TABLES` parameter of the `EXPDP` and `IMPDP` utilities.
  - Perform a full database export or import operation. If so, see the next guideline.

- **When performing an export or import operation for an entire database, set the EXPDP or IMPDP FULL option to Y.** Remember that this setting will capture the DVSYS schema, so ensure that the user or role has that you have authorized been granted the DV_OWNER role.

Note the following:

- You cannot use the legacy EXP and IMP utilities with the direct path option (direct=y) if Oracle Database Vault is enabled.
- Users, either through a direct grant or a role grant, that have been granted Database Vault-specific Oracle Data Pump authorization through the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure or transportable tablespace authorization through the DBMS_MACADM.AUTHORIZE_TTS_USER procedure can export and import database objects, but they cannot perform other activities, such as SELECT queries on schema tables to which they normally do not have access. Similarly, users are not permitted to perform Data Pump operations on objects outside the designated data objects.
- You must grant the DV_OWNER role to users who must export or import an entire database, because a full database export requires access to the DVSYS schema, which stores the Oracle Database Vault policies. However, you cannot export the DVSYS schema itself. Data Pump only exports the protection definitions. The target database must have the DVSYS schema in it and Database Vault enabled before you can begin the import process.) Conversely, for a Data Pump import operation to apply the imported policies to the target database, it internally uses the DBMS_MACADM PL/SQL package, which in turn requires the Data Pump user to have the DV_OWNER role.

> **See Also:**
>
> *Oracle Database Utilities* for detailed information about Oracle Data Pump

Using Oracle Scheduler with Oracle Database Vault

Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific authorization.

- About Using Oracle Scheduler with Oracle Database Vault
  The level of authorization that you must grant depends on the schema in which the administrator wants to perform a task.
- Granting a Job Scheduling Administrator Authorization for Database Vault
  You can authorize a user to schedule database jobs in a Database Vault environment.
- Revoking Authorization from Job Scheduling Administrators
  You can revoke authorization from a user for scheduling database jobs.

About Using Oracle Scheduler with Oracle Database Vault

The level of authorization that you must grant depends on the schema in which the administrator wants to perform a task.

Possible scenarios are as follows:

- **An administrator wants to schedule a job in his or her own schema.** An administrator who has been granted privileges to schedule database jobs can continue to do so without any Oracle Database Vault-specific authorizations, unless this schema is protected by a realm. In that case, ensure that this user is authorized to access the realm.
- **An administrator wants to run a job in another schema, but this job does not access any Oracle Database Vault realm or command rule protected object.** In this case, this user only needs job related system privileges, not the Oracle Database Vault privileges.
- **An administrator wants to run a job under the schema of another user, including any schema in the database or a remote database.** If this job accesses an Oracle Database Vault realm or command rule protected object, then you must grant this user Database Vault-specific authorization by using the `DBMS_MACADM.AUTHORIZE_SCHEDULER_USER` procedure. This authorization applies to both background and foreground jobs. For background jobs, the authorization applies to the last user who created or modified the job. In addition, ensure that the schema owner (the protected schema in which the job is created) authorized to the realm.

  Later on, you can revoke this authorization by using the `DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER` procedure. If the schema is not protected by a realm, then you do not need to run the `DBMS_MACADM.AUTHORIZE_SCHEDULER_USER` procedure for the user.

Related Topics

- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.

Granting a Job Scheduling Administrator Authorization for Database Vault

You can authorize a user to schedule database jobs in a Database Vault environment.

1. Log into the database instance as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant the necessary authorization.

2. Ensure that the user to whom you want to grant authorization has been granted system privileges to schedule database jobs.

   These privileges include any of the following: `CREATE JOB`, `CREATE ANY JOB`, `CREATE EXTERNAL JOB`, `EXECUTE ANY PROGRAM`, `EXECUTE ANY CLASS`, `MANAGE SCHEDULER`. The `DBA` and `SCHEDULER_ADMIN` roles provide these privileges; however, when Oracle Database Vault is enabled, the privileges are revoked from these roles.

   For example:

   ```
   SELECT GRANTEE, PRIVILEGE FROM DBA_SYS_PRIVS
    WHERE PRIVILEGE IN ('CREATE JOB', 'CREATE ANY JOB');
   ```

3. Grant this user Oracle Database Vault authorization.

For example, to authorize the user `job_mgr` to schedule jobs for any schema in the database:

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR');
```

Optionally, you can restrict `job_mgr`'s activities to a specific schema, as follows:

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

4. Ensure that the user has been authorized by querying the `DBA_DV_JOB_AUTH` data dictionary view as follows:

```
SELECT GRANTEE,SCHEMA FROM DBA_DV_JOB_AUTH WHERE GRANTEE = 'user_name';
```

Related Topics

- AUTHORIZE_SCHEDULER_USER Procedure
  The `AUTHORIZE_SCHEDULER_USER` procedure grants a user authorization to schedule database jobs when Oracle Database Vault is enabled.
- DBA_DV_JOB_AUTH View
  The `DBA_DV_JOB_AUTH` data dictionary view lists the authorizations for using Oracle Scheduler in an Oracle Database Vault environment.

Revoking Authorization from Job Scheduling Administrators

You can revoke authorization from a user for scheduling database jobs.

1. Query the `DBA_DV_JOB_AUTH` data dictionary view to find the user's authorization.

```
SELECT GRANTEE, SCHEMA FROM DBA_DV_JOB_AUTH WHERE GRANTEE='username';
```

2. Use the information you gathered from the preceding step to build the `DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER` command.

   For example:

```
EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR');
```

   Ensure that this unauthorization complements the original authorization action. In other words, if you originally gave `job_mgr` authorization over the entire database, then the following command will not work:

```
EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

Related Topics

- UNAUTHORIZE_SCHEDULER_USER Procedure
  The `UNAUTHORIZE_SCHEDULER_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_SCHEDULER_USER` procedure.

Using Information Lifecycle Management with Oracle Database Vault

Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.

About Using Information Lifecycle Management with Oracle Database Vault

You can grant authorization to and from users who are responsible for performing Information Lifecycle Management (ILM) operations on Oracle Database Vault realm- and command rule-protected objects.

You must first authorize users before they can perform the following SQL statements for ILM operations in a Database Vault-enabled database:

- `ALTER TABLE`
  - `ILM`
  - `FLASHBACK ARCHIVE`
  - `NO FLASHBACK ARCHIVE`
- `ALTER TABLESPACE`
  - `FLASHBACK MODE`

Authorizing Users for ILM Operations in Database Vault

You can authorize a user to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant the necessary authorization.

2. Use the `DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER` to authorize the user.

   For example, to grant a user authorization to perform ILM operations on the `HR.EMPLOYEES` table:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER ('PSMITH', 'HR', 'EMPLOYEES',
   'TABLE', 'ILM');
   ```

   If you wanted to grant user `psmith` ILM authorizations for the entire database, you would enter a procedure similar to the following:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER ('PSMITH', '%', '%', '%', '%');
   ```

3. Ensure that the user has been authorized by querying the `DBA_DV_MAINTENANCE_AUTH` data dictionary view.

Related Topics

- AUTHORIZE_MAINTENANCE_USER Procedure
  The AUTHORIZE_MAINTENANCE_USER procedure grants a user authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.
- DBA_DV_MAINTENANCE_AUTH View
  The DBA_DV_MAINTENANCE_AUTH data dictionary view provides information about the configuration of Oracle Database Vault authorizations to use Information Life Management (ILM) features.

Revoking Information Lifecycle Management Authorization from Users

You can revoke authorization from users so that they cannot perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

   Only a user who has been granted either of these roles can grant the necessary authorization.
2. Query the DBA_DV_MAINTENANCE_AUTH data dictionary view to find the kind of authorization that was granted to the ILM user.
3. Use the DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER to revoke the authorization from the user.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER ('PSMITH', 'HR', '%', 'TABLE',
   'ILM');
   ```

Related Topics

- DBA_DV_MAINTENANCE_AUTH View
  The DBA_DV_MAINTENANCE_AUTH data dictionary view provides information about the configuration of Oracle Database Vault authorizations to use Information Life Management (ILM) features.
- UNAUTHORIZE_MAINTENANCE_USER Procedure
  The UNAUTHORIZE_MAINTENANCE_USER procedure revokes privileges from users who have been granted authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

Using Oracle Database Replay with Oracle Database Vault

Database administrators can authorize Oracle Database Replay users to work in a Database Vault environment.

- About Using Database Replay with Oracle Database Vault
  You can grant Database Vault authorizations for users to perform both workload capture and workload replay operations with Oracle Database Replay.
- Authorizing Users for Database Replay Operations
  You can authorize Oracle Database Replay users for both workload capture and workload replay operations.

- Revoking Database Replay Authorization from Users
  You can remove authorization for both Oracle Database Replay workload capture and workload replay operations.

About Using Database Replay with Oracle Database Vault

You can grant Database Vault authorizations for users to perform both workload capture and workload replay operations with Oracle Database Replay.

Authorizing Users for Database Replay Operations

You can authorize Oracle Database Replay users for both workload capture and workload replay operations.

- Authorizing Users for Workload Capture Operations
  You can authorize a user to perform Oracle Database Replay workload capture operations in an Oracle Database Vault environment.
- Authorizing Users for Workload Replay Operations
  You can authorize a user to perform Oracle Database Replay workload replay operations in an Oracle Database Vault environment.

Authorizing Users for Workload Capture Operations

You can authorize a user to perform Oracle Database Replay workload capture operations in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant this authorization.

2. Use the `DBMS_MACADM.AUTHORIZE_DBCAPTURE` procedure to authorize the user.

   For example:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DBCAPTURE ('PFITCH');
   ```

3. Ensure that the user has been authorized by querying the `DBA_DV_DBCAPTURE_AUTH` data dictionary view.

Related Topics

- AUTHORIZE_DBCAPTURE Procedure
  The `AUTHORIZE_DBCAPTURE` procedure grants a user authorization to perform Oracle Database Replay workload capture operations.
- DBA_DV_DBCAPTURE_AUTH View
  The `DBA_DV_DBCAPTURE_AUTH` data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload capture operations.

Authorizing Users for Workload Replay Operations

You can authorize a user to perform Oracle Database Replay workload replay operations in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

   Only a user who has been granted either of these roles can grant this authorization.

2. Use the DBMS_MACADM.AUTHORIZE_DBREPLAY procedure to authorize the user.

   For example:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DBREPLAY ('PFITCH');
   ```

3. Ensure that the user has been authorized by querying the DBA_DV_DBREPLAY_AUTH data dictionary view.

Related Topics

- AUTHORIZE_DBREPLAY Procedure
  The AUTHORIZE_DBREPLAY procedure grants a user authorization to perform Oracle Database Replay workload replay operations.
- DBA_DV_DBREPLAY View
  The DBA_DV_DBREPLAY_AUTH data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload replay operations.

Revoking Database Replay Authorization from Users

You can remove authorization for both Oracle Database Replay workload capture and workload replay operations.

- Revoking Workload Capture Privileges
  You can revoke authorization from users so that they cannot perform Oracle Database Replay workload capture operations in an Oracle Database Vault environment.
- Revoking Workload Replay Privileges
  You can revoke authorization from users so that they cannot perform Oracle Database Replay workload replay operations in an Oracle Database Vault environment.

Revoking Workload Capture Privileges

You can revoke authorization from users so that they cannot perform Oracle Database Replay workload capture operations in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

   Only a user who has been granted either of these roles can grant this authorization.

2. Query the DBA_DV_DBCAPTURE_AUTH data dictionary view to find users whose workload capture authorization you want to revoke.

3. Use the DBMS_MACADM.UNAUTHORIZE_DBCAPTURE procedure to revoke authorization from the user.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_DBCAPTURE ('PFITCH');
   ```

Related Topics

- DBA_DV_DBCAPTURE_AUTH View
  The DBA_DV_DBCAPTURE_AUTH data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload capture operations.
- UNAUTHORIZE_DBCAPTURE Procedure
  The UNAUTHORIZE_DBCAPTURE procedure revokes authorization from users to perform Oracle Database Replay workload capture operations.

Revoking Workload Replay Privileges

You can revoke authorization from users so that they cannot perform Oracle Database Replay workload replay operations in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

   Only a user who has been granted either of these roles can grant this authorization.
2. Query the DBA_DV_DBREPLAY_AUTH data dictionary view to find users whose workload replay authorization you want to revoke.
3. Use the DBMS_MACADM.UNAUTHORIZE_DBDBREPLAY procedure to revoke authorization from the user.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_DBREPLAY ('PFITCH');
   ```

Related Topics

- DBA_DV_DBREPLAY View
  The DBA_DV_DBREPLAY_AUTH data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload replay operations.
- UNAUTHORIZE_DBREPLAY Procedure
  The UNAUTHORIZE_DBREPLAY procedure revokes authorization from users to perform Oracle Database Replay workload replay operations.

Executing Preprocessor Programs with Oracle Database Vault

Users who execute preprocessor programs through external tables must have Oracle Database Vault-specific authorization.

- About Executing Preprocessor Programs with Oracle Database Vault
  You can grant and revoke Database Vault authorizations for users to execute preprocessor programs through external tables.
- Authorizing Users to Execute Preprocessor Programs
  The DBMS_MACADM.AUTHORIZE_PREPROCESSOR procedure grants users authorization to execute preprocessor programs through external tables.

- Revoking Execute Preprocessor Authorization from Users
  The DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR procedure revokes authorization from users so that they cannot execute preprocessor programs through external tables in an Oracle Database Vault environment.

About Executing Preprocessor Programs with Oracle Database Vault

You can grant and revoke Database Vault authorizations for users to execute preprocessor programs through external tables.

Authorizing Users to Execute Preprocessor Programs

The DBMS_MACADM.AUTHORIZE_PREPROCESSOR procedure grants users authorization to execute preprocessor programs through external tables.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

   Only a user who has been granted either of these roles can grant this authorization.
2. Use the DBMS_MACADM.AUTHORIZE_PREPROCESSOR procedure to authorize the user.

   For example:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_PREPROCESSOR ('PFITCH');
   ```
3. Ensure that the user has been authorized by querying the DBA_DV_PREPROCESSOR_AUTH data dictionary view.

Revoking Execute Preprocessor Authorization from Users

The DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR procedure revokes authorization from users so that they cannot execute preprocessor programs through external tables in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

   Only a user who has been granted either of these roles can grant this authorization.
2. Use the DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR procedure to revoke the authorization from the user.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR ('PFITCH');
   ```
3. Query the DBA_DV_PREPROCESSOR_AUTH data dictionary view to ensure that the user is no longer authorized.

Oracle Recovery Manager and Oracle Database Vault

You can use Recovery Manager (RMAN) in an Oracle Database Vault environment.

The functionality of RMAN with Oracle Database Vault is the same as its functionality in a standard Oracle Database environment.

See Also:

- *Oracle Database Backup and Recovery User's Guide*
- *Oracle Database Backup and Recovery Reference*

Privileges for Using Oracle Streams with Oracle Database Vault

If you want to use Oracle Streams in an Oracle Database Vault environment, then you must have the correct privileges.

The privileges that you must have are as follows:

- You must be granted the `DV_STREAMS_ADMIN` role in order to configure the Oracle Streams capture process.
- Before you can apply changes to any tables that are protected by a realm, you must be authorized to have access to that realm. For example:

```
EXEC DBMS_MACADM.ADD_AUTH_TO_REALM('realm_name','username');
```

Related Topics

- DV_STREAMS_ADMIN Oracle Streams Configuration Role
  The `DV_STREAMS_ADMIN` role is used with Oracle Streams.
- ADD_AUTH_TO_REALM Procedure
  The `ADD_AUTH_TO_REALM` procedure authorizes a user or role to access a realm as an owner or a participant. In a multitenant environment, you can authenticate both common and local realms.

Privileges for Using XStream with Oracle Database Vault

If you want to use XStream in an Oracle Database Vault environment, then you must have the appropriate privileges.

These privileges are as follows:

- You must be granted the `DV_XSTREAM_ADMIN` role in order to configure the XStream.
- Before you can apply changes to any tables that are protected by a realm, you must be authorized to have access to that realm. For example:

```
EXEC DBMS_MACADM.ADD_AUTH_TO_REALM('realm_name','username');
```

Related Topics

- DV_XSTREAM_ADMIN XStream Administrative Role
  The `DV_XSTREAM_ADMIN` role is used for Oracle XStream.

- ADD_AUTH_TO_REALM Procedure
  The ADD_AUTH_TO_REALM procedure authorizes a user or role to access a realm as an owner or a participant. In a multitenant environment, you can authenticate both common and local realms.

Privileges for Using Oracle GoldenGate in with Oracle Database Vault

If you want to use Oracle GoldenGate in an Oracle Database Vault environment, then you must have the appropriate privileges.

These privileges are as follows:

- The user must be granted the DV_GOLDENGATE_ADMIN role in order to configure the Oracle GoldenGate.
- The user must be granted the DV_GOLDENGATE_REDO_ACCESS role if the user must use the Oracle GoldenGate TRANLOGOPTIONS DBLOGREADER method to access redo logs.

  For example, to grant the DV_GOLDENGATE_ADMIN and DV_GOLDENGATE_REDO_ACCESS roles to a user named gg_admin:

  ```
  GRANT DV_GOLDENGATE_ADMIN, DV_GOLDENGATE_REDO_ACCESS TO gg_admin;
  ```
- The user must be granted the DV_ACCTMGR role before this user can create users on the replicated side.
- The user must perform extract operations in triggerless mode before attempting to perform procedural replication.
- Before users can apply changes to any tables that are protected by a realm, they must be authorized to have access to that realm. For example:

  ```
  EXEC DBMS_MACADM.ADD_AUTH_TO_REALM('realm_name','username');
  ```
- The SYS user must be authorized to perform Data Definition Language (DDL) operations in the SYSTEM schema, as follows:

  ```
  EXECUTE DVSYS.DBMS_MACADM.AUTHORIZE_DDL('SYS', 'SYSTEM');
  ```
- The user must be granted authorization to the Oracle Default Component Protection Realm. For example, to grant this realm authorization to a user named gg_admin:

  ```
  BEGIN
   DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM(
     REALM_NAME    => 'Oracle Default Component Protection Realm',
     GRANTEE       => 'gg_admin',
     AUTH_OPTIONS  => 1);
  END;
  /
  ```

Related Topics

- DV_GOLDENGATE_ADMIN GoldenGate Administrative Role
  The DV_GOLDENGATE_ADMIN role is used with Oracle GoldenGate.
- DV_GOLDENGATE_REDO_ACCESS GoldenGate Redo Log Role
  The DV_GOLDENGATE_REDO_ACCESS role is used with Oracle GoldenGate.

- ADD_AUTH_TO_REALM Procedure
  The `ADD_AUTH_TO_REALM` procedure authorizes a user or role to access a realm as an owner or a participant. In a multitenant environment, you can authenticate both common and local realms.

Using Data Masking in an Oracle Database Vault Environment

You must have the correct authorization to perform data masking in an Oracle Database Vault environment.

- About Data Masking in an Oracle Database Vault Enabled Database
  In an Oracle Database Vault-enabled database, only users who have Database Vault authorizations can mask data in Database Vault-protected database objects.
- Adding Data Masking Users to the Data Dictionary Realm Authorizations
  You can add data masking users to the Oracle Default Component Protection realm to give them data dictionary realm authorizations.
- Giving Users Access to Tables or Schemas That They Want to Mask
  To give users access to tables or schemas that they want to mask, you must authorize them for the appropriate realm.
- Creating a Command Rule to Control Data Masking Privileges
  You must have privileges to manage tables, packages, and triggers before you can use data masking in an Oracle Database Vault environment.

About Data Masking in an Oracle Database Vault Enabled Database

In an Oracle Database Vault-enabled database, only users who have Database Vault authorizations can mask data in Database Vault-protected database objects.

In a non-Database Vault environment, users who have been granted the `SELECT_CATALOG_ROLE` and `DBA` roles can perform data masking. However, with Database Vault, users must have additional privileges. This section describes three ways that you can use to enable users to mask data in Database Vault-protected objects.

If users do not have the correct privileges, then the following errors can occur while creating the masking definition or when the job is executing:

```
ORA-47400: Command Rule violation for string on string

ORA-47401: Realm violation for string on string.

ORA-47408: Realm violation for the EXECUTE command

ORA-47409: Command Rule violation for the EXECUTE command

ORA-01301: insufficient privileges
```

Adding Data Masking Users to the Data Dictionary Realm Authorizations

You can add data masking users to the Oracle Default Component Protection realm to give them data dictionary realm authorizations.

The Oracle Data Dictionary controls access to the Oracle Database catalog schemas, such as SYS and SYSTEM. (See Default Realms for a full list of these schemas.) It also controls the ability to grant system privileges and database administrator roles. If you add users to the Oracle Default Component Protection realm, and assuming these users already have the privileges associated with the Oracle Data Dictionary, then these users will have these same privileges in a Database Vault environment. Therefore, if you do add a user to this realm, ensure that this user is a trusted user.

- To add a user to the Oracle Default Component Protection realm, use the DBMS_MACADM.ADD_AUTH_TO_REALM procedure.

For example:

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name   => 'Oracle Default Component Protection Realm',
  grantee      => 'DBA_JSMITH',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT);
END;
/
```

Giving Users Access to Tables or Schemas That They Want to Mask

To give users access to tables or schemas that they want to mask, you must authorize them for the appropriate realm.

If the table or schema of a table that is to be data masked is in a realm, then you must add the user responsible for data masking to the realm authorization as a participant or owner. If the table or schema has dependent objects that are in other realm-protected tables, then you must grant the user participant or owner authorization for those realms as well.

- To authorize users for data masking to a realm that protects the objects they want to data mask, use the DBMS_MACADM.ADD_AUTH_TO_REALM procedure.

The following example shows how to grant user DBA_JSMITH authorization for the HR.EMPLOYEES table, which is protected by a realm called Business Apps Realm:

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name   => 'Business Apps Realm',
  grantee      => 'DBA_JSMITH',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT;
END;
/
```

Creating a Command Rule to Control Data Masking Privileges

You must have privileges to manage tables, packages, and triggers before you can use data masking in an Oracle Database Vault environment.

For data masking, users must have the `CREATE TABLE`, `SELECT TABLE`, `ALTER TABLE`, and `DROP TABLE` privileges for the masking objects and if there are any dependent objects to be created, the user must have the appropriate privileges such as `CREATE PACKAGE`, `CREATE TRIGGER`, and so on.

You can create command rules to control data masking privileges at a granular level. To do so, create a command rule that can either prevent or allow the user access to objects that must have to be data masked. For example, you can create a command rule called Allow Data Masking that checks if the user is in a list of users who are responsible for data masking. If the user logging in is one of these users, then the command rule evaluates to true and the user is permitted to create the data mask for the protected object.

To create a command rule that controls data masking privileges:

1. Create the rule set rule.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Is HDRISCOLL or DBA_JSMITH User',
  rule_expr  =>'USER IN(''HDRISCOLL'',''DBA_JSMITH'')';
END;
/
```

2. Create a rule set and then add the rule to it:

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Allow Data Masking',
  description      => 'Allows users HDRISCOLL and DBA_JSMITH access',
  enabled          => 'Y',
  eval_options     => 1,
  audit_options    => 1,
  fail_options     => 1,
  fail_message     => 'You do not have access to this object.',
  fail_code        => 20461,
  handler_options  => 0,
  is_static        => TRUE);
END;
/
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Allow Data Masking',
  rule_name     => 'Is HDRISCOLL or DBA_JSMITH User'),
  rule_order    => 1);
END;
/
```

3. Create a command rule and then add this rule to it:

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'CREATE TABLE',
  rule_set_name   => 'Allow Data Masking',
```

```
  object_owner    => 'HR',
  object_name     => 'EMPLOYEES',
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB

You can convert a standalone Oracle Database Release 12c or later database to a PDB, and then plug this PDB into a CDB.

1. Connect to the root as a user who has been granted the `DV_OWNER` role.

   For example:

   ```
   sqlplus c##sec_admin
   Enter password: password
   ```

2. Grant the `DV_PATCH_ADMIN` role to user `SYS` with `CONTAINER = CURRENT`.

   ```
   GRANT DV_PATCH_ADMIN TO SYS CONTAINER = CURRENT;
   ```

3. In the root, connect as user `SYS` with the `SYSOPER` system privilege.

   For example:

   ```
   CONNECT SYS AS SYSOPER -- Or, CONNECT SYS@hrpdb AS SYSOPER
   Enter password: password
   ```

4. Restart the database in read-only mode.

   For example:

   ```
   SHUTDOWN IMMEDIATE
   STARTUP MOUNT
   ALTER DATABASE OPEN READ ONLY
   ```

5. Connect to the Database Vault-enabled database as a user who has the `DV_OWNER` role.

   For example:

   ```
   CONNECT sec_admin@dv_db
   ```

6. Grant the `DV_PATCH_ADMIN` role to user `SYS` on this database.

   ```
   GRANT DV_PATCH_ADMIN TO SYS;
   ```

7. Optionally, run the `DBMS_PDB.CHECK_PLUG_COMPATIBILITY` function to determine whether the unplugged PDB is compatible with the CDB.

   When you run the function, set the following parameters:

   - `pdb_descr_file`: Set this parameter to the full path to the XML file that will contain a description of the PDB.

- `store_report`: Set this parameter to indicate whether you want to generate a report if the PDB is not compatible with the CDB. Set it to `TRUE` to generate a report or `FALSE` to not generate a report. A generated report is stored in the `PDB_PLUG_IN_VIOLATIONS` temporary table and is generated only if the PDB is not compatible with the CDB.

For example, to determine whether a PDB described by the `/disk1/usr/dv_db_pdb.xml` file is compatible with the current CDB, run the following PL/SQL block:

```
SET SERVEROUTPUT ON
DECLARE
  compatible CONSTANT VARCHAR2(3) :=
    CASE DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
           pdb_descr_file => '/disk1/usr/dv_db_pdb.xml',
           store_report   => TRUE)
    WHEN TRUE THEN 'YES'
    ELSE 'NO'
END;
BEGIN
  DBMS_OUTPUT.PUT_LINE(compatible);
END;
/
```

If the output is `YES`, then the PDB is compatible, and you can continue with the next step.

If the output is `NO`, then the PDB is not compatible. You can check the `PDB_PLUG_IN_VIOLATIONS` temporary table to see why it is not compatible.

8. Create an XML file that describes the PDB.

   For example:

   ```
   BEGIN
     DBMS_PDB.DESCRIBE(
       pdb_descr_file => '/disk1/oracle/dv_db.xml');
   END;
   /
   ```

9. Run the `CREATE PLUGGABLE DATABASE` statement, and specify the XML file in the `USING` clause. Specify other clauses when they are required.

   For example:

   ```
   CREATE PLUGGABLE DATABASE dv_db_pdb AS CLONE USING 'dv_db.xml' NOCOPY;
   ```

10. Connect to the PDB that you just created as user `SYS` with the `SYSDBA` administrative privilege.

    ```
    CONNECT SYS@dv_db_pdb AS SYSDBA
    ```

11. Execute the `noncdb_to_pdb.sql` script.

    ```
    @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb.sql
    ```

12. Open this PDB in a read/write restricted mode.

```
ALTER PLUGGABLE DATABASE dv_db_pdb OPEN READ WRITE RESTRICTED;
```

13. Run the following procedure to synchronize the PDB:

```
EXECUTE DBMS_PDB.SYNC_PDB;
```

14. Connect to the root as a user who has been granted the DV_OWNER role.

```
sqlplus c##sec_admin
Enter password: password
```

15. Revoke the DV_PATCH_ADMIN role from user SYS with CONTAINER = CURRENT.

```
REVOKE DV_PATCH_ADMIN FROM SYS CONTAINER = CURRENT;
```

16. Connect to the legacy Database Vault-enabled database as user SYS with the SYSOPER system privilege.

```
CONNECT SYS@dv_db_pdb AS SYSOPER
```

17. Restart this database.

    For example:

```
SHUTDOWN IMMMEDIATE
STARUP
```

18. Revoke the DV_PATCH_ADMIN role from user SYS.

```
REVOKE DV_PATCH_ADMIN FROM SYS;
```

Using the ORADEBUG Utility with Oracle Database Vault

The ORADEBUG utility is used primarily by Oracle Support to diagnose problems that may arise with an Oracle database.

You can control whether users can run the ORADEBUG utility in an Oracle Database Vault-enabled environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. If necessary, find out if ORADEBUG is already disabled or enabled.

```
SELECT * FROM DBA_DV_ORADEBUG;
```

3. Run one of the following procedures:
   - To disable the use of ORADEBUG:

```
EXEC DBMS_MACADM.DISABLE_ORADEBUG;
```

   - To enable the use of ORADEBUG:

```
EXEC DBMS_MACADM.ENABLE_ORADEBUG;
```

Related Topics

- DBA_DV_ORADEBUG View
  The `DBA_DV_ORADEBUG` data dictionary view indicates whether users can use the `ORADEBUG` utility in an Oracle Database Vault environment.
- DISABLE_ORADEBUG Procedure
  The `DISABLE_ORADEBUG` procedure disables the use of the `ORADEBUG` utility in an Oracle Database Vault environment.
- ENABLE_ORADEBUG Procedure
  The `ENABLE_ORADEBUG` procedure enables the use of the `ORADEBUG` utility in an Oracle Database Vault environment.

14

Oracle Database Vault Schemas, Roles, and Accounts

Oracle Database Vault provides schemas that contain Database Vault objects, roles that provide separation of duty for specific tasks, and default user accounts.

- Oracle Database Vault Schemas
  The Oracle Database Vault schemas, DVSYS and DVF, support the administration and run-time processing of Oracle Database Vault.
- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.
- Oracle Database Vault Accounts Created During Registration
  You must create accounts for the Oracle Database Vault Owner and Oracle Database Vault Account Manager during the registration process.
- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the DV_OWNER and DV_ACCTMGR roles.

Oracle Database Vault Schemas

The Oracle Database Vault schemas, DVSYS and DVF, support the administration and run-time processing of Oracle Database Vault.

- DVSYS Schema
  The DVSYS schema contains Oracle Database Vault database objects.
- DVF Schema
  The DVF schema is the owner of the Oracle Database Vault DBMS_MACSEC_FUNCTION PL/SQL package.

DVSYS Schema

The DVSYS schema contains Oracle Database Vault database objects.

These objects store Oracle Database Vault configuration information and support the administration and run-time processing of Oracle Database Vault.

In a default installation, the DVSYS schema is locked. The DVSYS schema also owns the AUDIT_TRAIL$ table.

In a multitenant environment, the DVSYS schema is considered a common schema, which means that the objects within DVSYS (tables, views, PL/SQL packages, and so on) are automatically available to any child pluggable databases (PDBs). In addition, the DVSYS schema account cannot switch to other containers using the ALTER SESSION statement.

Oracle Database Vault secures the DVSYS schema by using a protected schema design. A protected schema design guards the schema against improper use of system privileges (for example, SELECT ANY TABLE, CREATE ANY VIEW, or DROP ANY).

Oracle Database Vault protects and secures the DVSYS schema in the following ways:

- The DVSYS protected schema and its administrative roles cannot be dropped. By default, the DVSYS account is locked.
- By default, users cannot directly log into the DVSYS account. To control the ability of users to directly log into this account, you can run the DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS procedure to prevent users from logging in and the DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS procedure to allow users to log in.
- Statements such as CREATE USER, ALTER USER, DROP USER, CREATE PROFILE, ALTER PROFILE, and DROP PROFILE can only be issued by a user with the DV_ACCTMGR role. A user logged in with the SYSDBA administrative privilege can issue these statements only if it is allowed to do so by modifying the Can Maintain Accounts/ Profiles rule set.
- The powerful ANY system privileges for database definition language (DDL) and data manipulation language (DML) commands are blocked in the protected schema. This means that the objects in the DVSYS schema must be created by the schema account itself. Also, access to the schema objects must be authorized through object privilege grants.
- Object privileges in the DVSYS schema can only be granted to Database Vault administrative roles in the schema. This means that users can access the protected schema only through predefined administrative roles.
- Only the protected schema account DVSYS can issue ALTER ROLE statements on Database Vault predefined administrative roles of the schema. Oracle Database Vault Roles describes Oracle Database Vault predefined administrative roles in detail.
- The SYS.DBMS_SYS_SQL.PARSE_AS_USER procedure cannot be used to run SQL statements on behalf of the protected schema DVSYS.

> **Note:**
>
> Database users can grant additional object privileges and roles to the Oracle Database Vault administrative roles (DV_ADMIN and DV_OWNER, for example) provided they have sufficient privileges to do so.

DVF Schema

The DVF schema is the owner of the Oracle Database Vault DBMS_MACSEC_FUNCTION PL/SQL package.

This package contains the functions that retrieve factor identities. After you install Oracle Database Vault, the installation process locks the DVF account to better secure it. When you create a new factor, Oracle Database Vault creates a new retrieval function for the factor and saves it in this schema.

In a multitenant environment, the DVF user cannot switch to other containers using the ALTER SESSION statement.

By default, users cannot directly log into the DVF account. To control the ability of users to directly log into this account, you can run the DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS procedure to prevent users from logging in and the DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS procedure to allow users to log in.

Oracle Database Vault Roles

Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.

- About Oracle Database Vault Roles
  Oracle Database Vault provides a set of roles that are required for managing Oracle Database Vault.
- Privileges of Oracle Database Vault Roles
  The Oracle Database Vault roles are designed to provide the maximum benefits of separation of duty.
- Granting Oracle Database Vault Roles to Users
  You can use Enterprise Manager Cloud Control to grant Oracle Database Vault roles to users.
- DV_OWNER Database Vault Owner Role
  The DV_OWNER role enables you to manage the Oracle Database Vault roles and its configuration.
- DV_ADMIN Database Vault Configuration Administrator Role
  The DV_ADMIN role controls the Oracle Database Vault PL/SQL packages.
- DV_MONITOR Database Vault Monitoring Role
  The DV_MONITOR role is used for monitoring Oracle Database Vault.
- DV_SECANALYST Database Vault Security Analyst Role
  The DV_SECANALYST role enables users to analyze activities.
- DV_AUDIT_CLEANUP Audit Trail Cleanup Role
  The DV_AUDIT_CLEANUP role is used for purge operations.
- DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role
  The DV_DATAPUMP_NETWORK_LINK role is used for Data Pump import operations.
- DV_STREAMS_ADMIN Oracle Streams Configuration Role
  The DV_STREAMS_ADMIN role is used with Oracle Streams.
- DV_XSTREAM_ADMIN XStream Administrative Role
  The DV_XSTREAM_ADMIN role is used for Oracle XStream.
- DV_GOLDENGATE_ADMIN GoldenGate Administrative Role
  The DV_GOLDENGATE_ADMIN role is used with Oracle GoldenGate.
- DV_GOLDENGATE_REDO_ACCESS GoldenGate Redo Log Role
  The DV_GOLDENGATE_REDO_ACCESS role is used with Oracle GoldenGate.
- DV_PATCH_ADMIN Database Vault Database Patch Role
  The DV_PATCH_ADMIN role is used for patching operations.

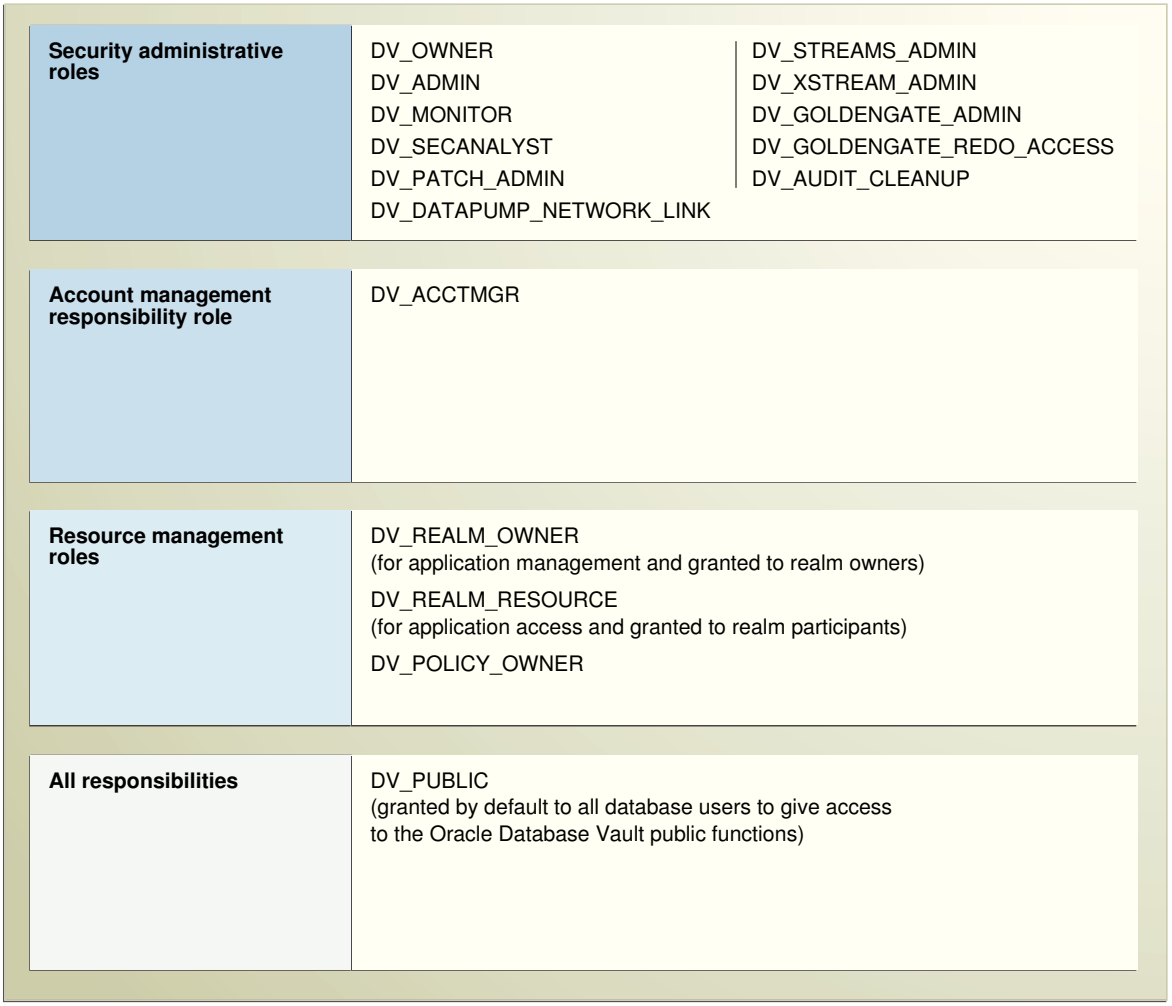
- DV_ACCTMGR Database Vault Account Manager Role
  The DV_ACCTMGR role is a powerful role, used for accounts management.
- DV_REALM_OWNER Database Vault Realm DBA Role
  The DV_REALM_OWNER role is used for realm management.
- DV_REALM_RESOURCE Database Vault Application Resource Owner Role
  The DV_REALM_RESOURCE role is use for the management of realm resources.
- DV_POLICY_OWNER Database Vault Owner Role
  The DV_POLICY_OWNER role enables database users to manage to a limited degree Oracle Database Vault policies.
- DV_PUBLIC Database Vault PUBLIC Role
  The DV_PUBLIC role is no longer used.

About Oracle Database Vault Roles

Oracle Database Vault provides a set of roles that are required for managing Oracle Database Vault.

Figure 14-1 illustrates how these roles are designed to implement the first level of separation of duties within the database. How you use these roles depends on the requirements that your company has in place.

Figure 14-1 How Oracle Database Vault Roles Are Categorized



Note:

You can grant additional object privileges and roles to the Oracle Database Vault roles to extend their scope of privileges. For example, a user logged in with the `SYSDBA` administrative privilege can grant object privileges to an Oracle Database Vault role as long as the object is not in the `DVSYS` schema or realm.

See Also:

- Separation of Duty Guidelines
- Managing Oracle Database Administrative Accounts
- *Oracle Database Security Guide*

Privileges of Oracle Database Vault Roles

The Oracle Database Vault roles are designed to provide the maximum benefits of separation of duty.

The `DV_PATCH_ADMIN`, `DV_STREAMS_ADMIN`, `DV_XSTREAM`, `DV_GOLDENGATE_ADMIN`, and `DV_GOLDENGATE_REDO_ACCESS` roles are not included in the following table because they have no system privileges.

Table 14-1 summarizes the privileges available with Oracle Database Vault roles.

Table 14-1 Privileges of Oracle Database Vault Roles

Privilege	DV_OWNER	DV_ADMIN	DV_MONITOR	DV_SECANALYST	DV_ACCTMGR	DV_REALM_OWNER	DV_REALM_RESOURCE	DV_PUBLIC	DV_POLICY_OWNER	DV_AUDIT_CLEANUP
`DVSYS` schema, `EXECUTE`	Yes[1]	Yes[2]	No	No	No	No	No	No	Yes, on some `DBMS_MACADM` procedures	No
`DVSYS` schema, `SELECT`	Yes	Yes	Yes	Yes, on some Database Vault views[3]	No	No	No	No[4]	Yes, on some Database Vault views[5]	Yes, on some Database Vault tables and views[6]
`DVSYS` schema, `DELETE`	Yes, on some Database Vault tables and views [7]	No	No	No	No	No	No	No	No	Yes, on some Database Vault tables and views [8]
`DVSYS` schema, grant privileges on objects	No	No	No	No	No	No	No	No	No	No

Table 14-1 (Cont.) Privileges of Oracle Database Vault Roles

Privilege	DV_OWNER	DV_ADMIN	DV_MONITOR	DV_SECANALYST	DV_ACCTMGR	DV_REALM_OWNER	DV_REALM_RESOURCE	DV_PUBLIC	DV_POLICY_OWNER	DV_AUDIT_CLEANUP
DVF schema, EXECUTE	Yes	No	No	No	No	No	No	No	No	No
DVF schema, SELECT	Yes	No	No	Yes	No	No	No	No	No	No
Monitor Database Vault	Yes	Yes	Yes	Yes	No	No	No	No	No	No
Run Database Vault reports	Yes	Yes	No	Yes	No	No	No	No	No	No
SYS schema, SELECT	Yes	No	Yes	Yes, on the same system views as DV_OWNER and DV_ADMIN	No	No	No	No	No	No
SYSMAN schema, SELECT	Yes, portions of	No	No	Yes, portions of	No	No	No	No	No	No
CREATE, ALTER, DROP user accounts and profiles[9]	No	No	No	No	Yes	No	No	No	No	No
Manage objects in schemas that define a realm[10]	No	No	No	No	No	Yes[11]	No	No	No	No
RESOURCE role privileges[12]	No	No	No	No	No	No	Yes	No	No	No

1 Includes the EXECUTE privilege on all Oracle Database Vault PL/SQL packages.
2 Includes the EXECUTE privilege on all Oracle Database Vault PL/SQL packages.
3 DV_SECANALYST can query DVSYS schema objects through Oracle Database Vault-supplied views only.
4 DV_PUBLIC can query DVSYS schema objects through Oracle Database Vault-supplied views only.
5 POLICY_OWNER* views only

6 DV_AUDIT_CLEANUP can perform SELECT statements on the AUDIT_TRAIL$ table and the DV$ENFORCEMENT_AUDIT, and DV$CONFIGURATION_AUDIT views.
7 DV_AUDIT_CLEANUP can perform DELETE statements on the AUDIT_TRAIL$ table and the DV$ENFORCEMENT_AUDIT, and DV$CONFIGURATION_AUDIT views.
8 DV_AUDIT_CLEANUP can perform DELETE statements on the AUDIT_TRAIL$ table and the DV$ENFORCEMENT_AUDIT, and DV$CONFIGURATION_AUDIT views.
9 This privilege does not include the ability to drop or alter the DVSYS account, nor change the DVSYS password.
1 This privilege includes ANY privileges, such as CREATE ANY, ALTER ANY, and DROP ANY.
0
1 The user with this role also must be the realm participant or owner to exercise his or her system privileges.
1
1 The RESOURCE role provides the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE
2 OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE.

Granting Oracle Database Vault Roles to Users

You can use Enterprise Manager Cloud Control to grant Oracle Database Vault roles to users.

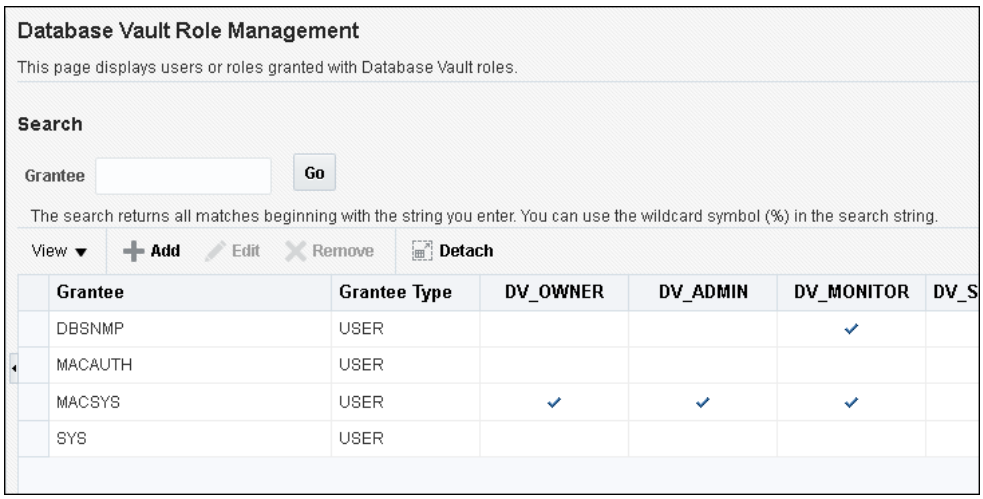
1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the DV_OWNER role and the SELECT ANY DICTIONARY privilege..

   Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.

   Refer to the role descriptions to find the requirements for who can grant roles to other users.

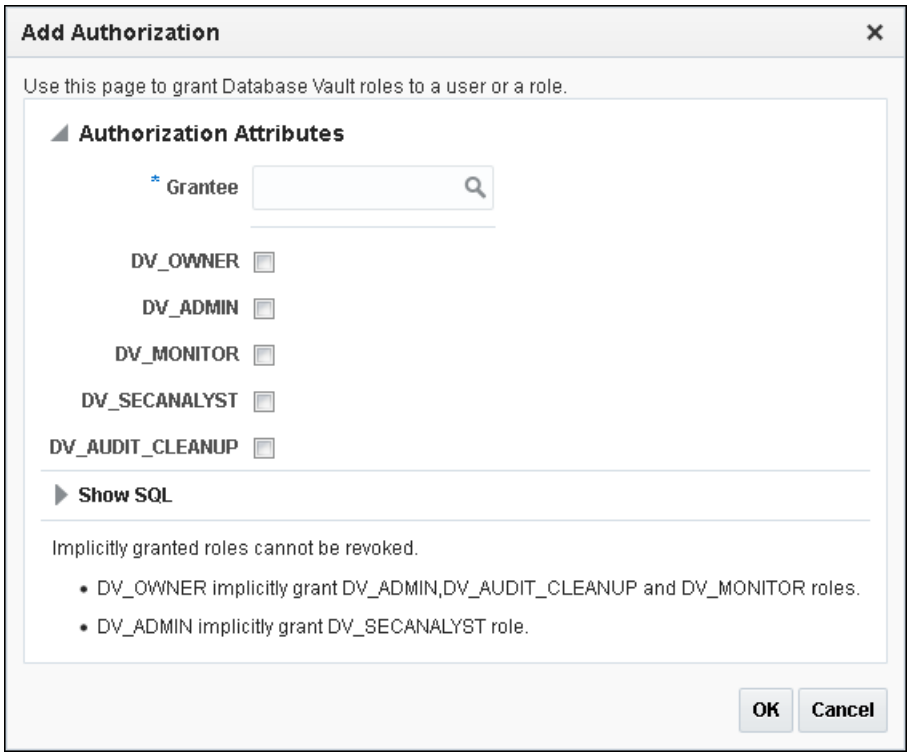
2. In the Administration page, under Database Vault Components, click **Database Vault Role Management**.

   The Database Vault Role Management page appears.



3. Do one of the following:

   - To add a new user or role for a grant, click the Add button to display the Add Authorization dialog box. Enter the grantee in the **Grantee** field, and then select the roles for the grant. Then click **OK**.



- To grant different roles or modify role grants for a user or role listed in the Database Vault Role Management page, select the user or role, click **Edit**, and then modify the role grants as necessary. Then click **OK**.

DV_OWNER Database Vault Owner Role

The `DV_OWNER` role enables you to manage the Oracle Database Vault roles and its configuration.

In *Oracle Database Vault Administrator's Guide*, the example account that uses this role is `leo_dvowner`.

Privileges Associated with the DV_OWNER Role

The `DV_OWNER` role has the administrative capabilities that the `DV_ADMIN` role provides, and the reporting capabilities the `DV_SECANALYST` role provides.

This role also provides privileges for monitoring Oracle Database Vault. It is created when you install Oracle Database Vault, and has the most privileges on the `DVSYS` schema. It also has the `DV_ADMIN` role.

To find the full list of system and object privileges associated with the `DV_OWNER` role, you can log into the database instance and enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_OWNER';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_OWNER';
```

When you install and register Oracle Database Vault, the `DV_OWNER` account is created. The user who is granted this role is also granted the `ADMIN` option and can grant any Oracle Database Vault roles (except `DV_ACCTMGR`) to any account. Users granted this role also can run Oracle Database Vault reports and monitor Oracle Database Vault.

> **Tip:**
>
> Oracle strongly recommends that you create separate, named account for the DV_OWNER user. This way, if the user is no longer available (for example, he or she left the company), then you can easily recreate this user account and then grant this user the DV_OWNER role.

How Are GRANT and REVOKE Operations Affected by DV_OWNER?

Anyone with the DV_OWNER role can grant the DV_OWNER and DV_ADMIN roles to another user.

The account granted this role can revoke any granted Database Vault role from another account. Accounts such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone (directly granted or indirectly granted using a role) do not have the right to grant or revoke the DV_OWNER role to or from any other database account. Note also that a user with the DV_OWNER role cannot grant or revoke the DV_ACCTMGR role.

Managing Password Changes for Users Who Have the DV_OWNER Role

Before you can change the password for another user who has been granted the DV_OWNER role, you must revoke the DV_OWNER role from that user account.

However, be cautious about revoking the DV_OWNER role. At least one user on your site must have this role granted. If another DV_OWNER user has been granted this role and needs to have his or her password changed, then you can temporarily revoke DV_OWNER from that user. Note also that if you have been granted the DV_OWNER role, then you can change your own password without having to revoke the role from yourself.

To change the DV_OWNER user password:

1. Log into the database instance using an account that has been granted the DV_OWNER role.
2. Revoke the DV_OWNER role from the user account whose password needs to change.
3. Connect as a user who has been granted the DV_ACCTMGR role and then change the password for this user.
4. Connect as the DV_OWNER user and then grant the DV_OWNER role back to the user whose password you changed.

DV_OWNER Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_ADMIN Database Vault Configuration Administrator Role

The DV_ADMIN role controls the Oracle Database Vault PL/SQL packages.

These packages are the underlying interface for the Database Vault Administrator user interface in Oracle Enterprise Manager Cloud Control.

Privileges Associated with the DV_ADMIN Role

The DV_ADMIN role has the EXECUTE privilege on the DVSYS packages (DBMS_MACADM, DBMS_MACSECROLES, and DBMS_MACUTL).

DV_ADMIN also has the capabilities provided by the DV_SECANALYST role, which allow the user to run Oracle Database Vault reports and monitor Oracle Database Vault. During installation, the DV_ADMIN role is granted to the DV_OWNER role with the ADMIN OPTION.

In addition, the DV_ADMIN role provides the SELECT privilege on the DBA_DV_POLICY, DBA_DV_POLICY_OWNER, and DBA_DV_POLICY_OBJECT data dictionary views. The DV_ADMIN role also has the REGISTER SESSION system privilege.

To find the full list of system and object privileges associated with the DV_ADMIN role, log into the database instance with sufficient privileges and then enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_ADMIN';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_ADMIN';
```

How Are GRANT and REVOKE Operations Affected by DV_ADMIN?

Accounts such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone do not have the rights to grant or revoke DV_ADMIN from any other database account.

The user with the DV_OWNER role can grant or revoke this role to and from any database account.

Managing Password Changes for Users Who Have the DV_ADMIN Role

Before you can change the password for a user who has been granted the DV_ADMIN role, you must revoke the DV_ADMIN role from this account.

If you have been granted the DV_ADMIN role, then you can change your own password without having to revoke the role from yourself.

To change the DV_ADMIN user password:

1. Log into the database instance using an account that has been granted the DV_OWNER role.
2. Revoke the DV_ADMIN role from the user account whose password needs to change.
3. Connect as a user who has been granted the DV_ACCTMGR role and then change the password for this user.

4. Connect as the DV_OWNER user and then grant the DV_ADMIN role back to the user whose password you changed.

DV_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_MONITOR Database Vault Monitoring Role

The DV_MONITOR role is used for monitoring Oracle Database Vault.

The DV_MONITOR role enables the Oracle Enterprise Manager Cloud Control agent to monitor Oracle Database Vault for attempted violations and configuration issues with realm or command rule definitions.

This role enables Cloud Control to read and propagate realm definitions and command rule definitions between databases.

Privileges Associated with the DV_MONITOR Role

There are no system privileges associated with the DV_MONITOR role, but it does have the SELECT privilege on SYS and DVSYS objects.

In addition, the DV_MONITOR role provides the SELECT privilege on the DBA_DV_POLICY, DBA_DV_POLICY_OWNER, and DBA_DV_POLICY_OBJECT data dictionary views.

To find the full list of DV_MONITOR object privileges, log into the database instance with sufficient (such as DV_OWNER) privileges and then enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_MONITOR';
```

How Are GRANT and REVOKE Operations Affected by DV_MONITOR?

By default, the DV_MONITOR role is granted to the DV_OWNER role and the DBSNMP user.

Only a user who has been granted the DV_OWNER role can grant or revoke the DV_MONITOR role to another user.

DV_MONITOR Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Monitoring Oracle Database Vault
  You can monitor Oracle Database Vault by checking for violations to the Database Vault configurations and by tracking changes to policies.
- Auditing Oracle Database Vault
  You can audit activities in Oracle Database Vault, such as changes to policy configurations.
- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_SECANALYST Database Vault Security Analyst Role

The DV_SECANALYST role enables users to analyze activities.

Use the DV_SECANALYST role to run Oracle Database Vault reports and monitor Oracle Database Vault.

This role is also used for database-related reports. In addition, this role enables you to check the DVSYS configuration by querying the DVSYS views described in Oracle Database Vault Data Dictionary Views.

Privileges Associated with the DV_SECANALYST Role

There are no system privileges associated with the DV_SECANALYST role, but it does have the SELECT privilege for some DVSYS schema objects and portions of the SYS and SYSMAN schema objects for reporting on DVSYS- and DVF-related entities.

In addition, the DV_SECANALYST role provides the SELECT privilege on the DBA_DV_POLICY, DBA_DV_POLICY_OWNER, and DBA_DV_POLICY_OBJECT data dictionary views.

To find the full list of DV_SECANALYST object privileges, log into the database instance with sufficient privileges and then enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE =
'DV_SECANALYST';
```

How Are GRANT and REVOKE Operations Affected by DV_SECANALYST?

Any account, such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone does not have the rights to grant this role to or revoke this role from any other database account.

Only the user with the DV_OWNER role can grant or revoke this role to and from another user.

DV_SECANALYST Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_AUDIT_CLEANUP Audit Trail Cleanup Role

The DV_AUDIT_CLEANUP role is used for purge operations.

> **Note:**
>
> This feature has been updated in Oracle Database 12*c* release 1 (12.1.0.2).

Grant the DV_AUDIT_CLEANUP role to any user who is responsible for purging the Database Vault audit trail in a non-unified auditing environment.

Archiving and Purging the Oracle Database Vault Audit Trail explains how to use this role to complete a purge operation.

Privileges Associated with the DV_AUDIT_CLEANUP Role

The DV_AUDIT_CLEANUP role has SELECT and DELETE privileges for three Database Vault-related auditing views.

- SELECT and DELETE on the DVSYS.AUDIT_TRAIL$ table
- SELECT and DELETE on the DVSYS.DV$ENFORCEMENT_AUDIT view
- SELECT and DELETE on the DVSYS.DV$CONFIGURATION_AUDIT view

How Are GRANT and REVOKE Operations Affected by DV_AUDIT_CLEANUP?

By default, this role is granted to the DV_OWNER role with the ADMIN OPTION.

Only a user who has been granted the DV_OWNER role can grant or revoke the DV_AUDIT_CLEANUP role to another user.

DV_AUDIT_CLEANUP Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role

The DV_DATAPUMP_NETWORK_LINK role is used for Data Pump import operations.

Grant the DV_DATAPUMP_NETWORK_LINK role to any user who is responsible for conducting the NETWORK_LINK transportable Data Pump import operation in an Oracle Database Vault environment.

This role enables the management of the Oracle Data Pump NETWORK_LINK transportable import processes to be tightly controlled by Database Vault, but does not change or restrict the way you would normally conduct Oracle Data Pump operations.

Privileges Associated with the DV_DATAPUMP_NETWORK_LINK Role

There are no system privileges associated with the DV_DATAPUMP_NETWORK_LINK role, but it does have the EXECUTE privilege on DVSYS objects.

To find the full list of DV_DATAPUMP_NETWORK_LINK object privileges, log into the database instance with sufficient privileges and then enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE =
'DV_DATAPUMP_NETWORK_LINK';
```

Be aware that the DV_DATAPUMP_NETWORK_LINK role does not provide a sufficient set of database privileges to conduct NETWORK_LINK transportable Data Pump import operation. Rather, the DV_DATAPUMP_NETWORK_LINK role is an additional requirement (that is, in addition to the privileges that Oracle Data Pump currently requires) for database administrators to conduct NETWORK_LINK transportable Data Pump import operations in an Oracle Database Vault environment.

How Are GRANT and REVOKE Operations Affected by DV_DATAPUMP_NETWORK_LINK?

Only users who have been granted the DV_OWNER role can grant or revoke the DV_DATAPUMP_NETWORK_LINK role to or from other users.

DV_DATAPUMP_NETWORK_LINK Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Using Oracle Data Pump with Oracle Database Vault
  Database administrators can authorize Oracle Data Pump users to work in a Database Vault environment.
- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_STREAMS_ADMIN Oracle Streams Configuration Role

The DV_STREAMS_ADMIN role is used with Oracle Streams.

Grant the DV_STREAMS_ADMIN role to any user who is responsible for configuring Oracle Streams in an Oracle Database Vault environment.

This enables the management of Oracle Streams processes to be tightly controlled by Database Vault, but does not change or restrict the way an administrator would normally configure Oracle Streams.

Privileges Associated with the DV_STREAMS_ADMIN Role

There are no system privileges associated with the DV_STREAMS_ADMIN role, but it does have the SELECT privilege on DVSYS objects.

To find the full list of DV_STREAMS_ADMIN object privileges, log into the database instance with sufficient privileges and then enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE =
'DV_STREAMS_ADMIN';
```

Be aware that the DV_STREAMS_ADMIN role does not provide a sufficient set of database privileges for configuring Oracle Streams. Rather, the DV_STREAMS_ADMIN role is an additional requirement (that is, in addition to the privileges that Oracle Streams currently requires) for database administrators to configure Oracle Streams in an Oracle Database Vault environment.

How Are GRANT and REVOKE Operations Affected by DV_STREAMS_ADMIN?

Only users who have been granted the DV_OWNER role can grant or revoke the DV_STREAMS_ADMIN role to or from other users.

DV_STREAMS_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_XSTREAM_ADMIN XStream Administrative Role

The DV_XSTREAM_ADMIN role is used for Oracle XStream.

Grant the DV_XSTREAM_ADMIN role to any user who is responsible for configuring Oracle XStream in an Oracle Database Vault environment.

This enables the management of XStream processes to be tightly controlled by Database Vault, but does not change or restrict the way an administrator would normally configure XStream.

Privileges Associated with the DV_XSTREAM_ADMIN Role

There are no privileges associated with the DV_XSTREAM_ADMIN role.

Be aware that the DV_XSTREAM_ADMIN role does not provide a sufficient set of database privileges for configuring XStream. Rather, the DV_XSTREAM_ADMIN role is an additional requirement (that is, in addition to the privileges that XStream currently requires) for

database administrators to configure XStream in an Oracle Database Vault environment.

How Are GRANT and REVOKE Operations Affected by DV_XSTREAM_ADMIN?

Only users who have been granted the `DV_OWNER` role can grant or revoke the `DV_XSTREAM_ADMIN` role to or from other users.

DV_XSTREAM_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the `GRANT ANY ROLE` system privilege can perform `GRANT` and `REVOKE` operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.
- Privileges for Using XStream with Oracle Database Vault
  If you want to use XStream in an Oracle Database Vault environment, then you must have the appropriate privileges.

DV_GOLDENGATE_ADMIN GoldenGate Administrative Role

The `DV_GOLDENGATE_ADMIN` role is used with Oracle GoldenGate.

Grant theto any user who is responsible for configuring Oracle GoldenGate in an Oracle Database Vault environment.

This enables the management of Oracle GoldenGate processes to be tightly controlled by Database Vault, but does not change or restrict the way an administrator would normally configure Oracle GoldenGate.

Privileges Associated with the DV_GOLDENGATE_ADMIN Role

There are no privileges associated with the `DV_GOLDENGATE_ADMIN` role.

Be aware that the `DV_GOLDENGATE_ADMIN` role does not provide a sufficient set of database privileges for configuring Oracle GoldenGate. Rather, the `DV_GOLDENGATE_ADMIN` role is an additional requirement (that is, in addition to the privileges that Oracle GoldenGate currently requires) for database administrators to configure Oracle GoldenGate in an Oracle Database Vault environment.

How Are GRANT and REVOKE Operations Affected by DV_GOLDENGATE_ADMIN?

Only users who have been granted the `DV_OWNER` role can grant or revoke the `DV_GOLDENGATE_ADMIN` role to or from other users.

DV_GOLDENGATE_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.
- Privileges for Using Oracle GoldenGate in with Oracle Database Vault
  If you want to use Oracle GoldenGate in an Oracle Database Vault environment, then you must have the appropriate privileges.

DV_GOLDENGATE_REDO_ACCESS GoldenGate Redo Log Role

The DV_GOLDENGATE_REDO_ACCESS role is used with Oracle GoldenGate.

Grant the DV_GOLDENGATE_REDO_ACCESS role to any user who is responsible for using the Oracle GoldenGate TRANLOGOPTIONS DBLOGREADER method to access redo logs in an Oracle Database Vault environment.

This enables the management of Oracle GoldenGate processes to be tightly controlled by Database Vault, but does not change or restrict the way an administrator would normally configure Oracle GoldenGate.

Privileges Associated with the DV_GOLDENGATE_REDO_ACCESS Role

There are no privileges associated with the DV_GOLDENGATE_REDO_ACCESS role.

Be aware that the DV_GOLDENGATE_REDO_ACCESS role does not provide a sufficient set of database privileges for configuring Oracle GoldenGate. Rather, the DV_GOLDENGATE_REDO_ACCESS role is an additional requirement (that is, in addition to the privileges that Oracle GoldenGate currently requires) for database administrators to configure Oracle Streams in an Oracle Database Vault environment.

How Are GRANT and REVOKE Operations Affected by DV_GOLDENGATE_REDO_ACCESS?

You cannot grant the DV_GOLDENGATE_REDO_ACCESS role with ADMIN OPTION.

Only users who have been granted the DV_OWNER role can grant or revoke the DV_GOLDENGATE_REDO_ACCESS role to or from other users.

DV_GOLDENGATE_REDO_ACCESS Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

- Privileges for Using Oracle GoldenGate in with Oracle Database Vault
  If you want to use Oracle GoldenGate in an Oracle Database Vault environment, then you must have the appropriate privileges.

DV_PATCH_ADMIN Database Vault Database Patch Role

The DV_PATCH_ADMIN role is used for patching operations.

In order to generate all Database Vault-related audit records in accordance with the audit policies specified in the Database Vault metadata as well as Database Vault unified audit policies, execute the DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT procedure as a user who has been granted the DV_ADMIN role before using the DV_PATCH_ADMIN role.

Temporarily grant the DV_PATCH_ADMIN role to any database administrator who is responsible for performing database patching. Before this administrator performs the patch operation, run the DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT procedure. This procedure enables realm, command rule, and rule set auditing of the actions by users who have been granted the DV_PATCH_ADMIN role, in accordance with the existing audit configuration. If you have mixed-mode auditing, then this user's actions are written to the AUDIT_TRAIL$ table. If you have pure unified auditing enabled, then you should create a unified audit policy to capture this user's actions.

After the patch operation is complete, do not immediately disable the auditing of users who are responsible for performing database patch operations. This way, you can track the actions of the DV_PATCH_ADMIN role users. For backwards compatibility, this type of auditing is disabled by default.

Privileges Associated with the DV_PATCH_ADMIN Role

The DV_PATCH_ADMIN role does not provide access to any secured data.

The DV_PATCH_ADMIN role a special Database Vault role that does not have any object or system privilege. It is designed to allow the database administrator or the user SYS to patch Database Vault enabled databases (for example, applying a database patch without disabling Database Vault). It also enables the database administrator to create users, because some patches may require the need to create new schemas.

How Are GRANT and REVOKE Operations Affected by DV_PATCH_ADMIN?

Only a user who has the DV_OWNER role can grant or revoke the DV_PATCH_ADMIN role to and from another user.

DV_PATCH_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

> **See Also:**
>
> - *Oracle Database Security Guide* for information about creating unified audit policies
> - Disabling and Enabling Oracle Database Vault

DV_ACCTMGR Database Vault Account Manager Role

The DV_ACCTMGR role is a powerful role, used for accounts management.

Use the DV_ACCTMGR role to create and maintain database accounts and database profiles. In this manual, the example DV_ACCTMGR role is assigned to a user named bea_dvacctmgr.

Privileges Associated with the DV_ACCTMGR Role

A user who has been granted this role can use the CREATE, ALTER, and DROP statements for user accounts or profiles, including users who have been granted the DV_SECANALYST, DV_AUDIT_CLEANUP, and DV_MONITOR roles.

This user also can grant the CREATE SESSION privilege to other users. However, a person who has been granted the DV_ACCTMGR role cannot perform the following operations:

- ALTER or DROP statements on the DVSYS account
- ALTER or DROP statements on users who have been granted the DV_ADMIN or DV_OWNER role
- Change passwords for users who have been granted the DV_ADMIN or DV_OWNER role

A common user who has been granted the DV_ACCTMGR role in the CDB root can alter a common user or a common profile in the CDB root even if the common DV_ACCTMGR user does not have the SET CONTAINER privilege or the DV_ACCTMGR role in any PDB.

To find the full list of system and object privileges associated with the DV_ACCTMGR role, log into the database instance with sufficient privileges and then enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_ACCTMGR';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_ACCTMGR';
```

Tips:

- If you want the DV_ACCTMGR user to be able to grant or revoke the ANY privileges for other users, then log in as user SYS with the SYSDBA privilege and grant this user the GRANT ANY PRIVILEGE and REVOKE ANY PRIVILEGE privileges. Then add this user to the Oracle System Privilege and Role Management Realm as an owner.
- Oracle strongly recommends that you create a separate, named account for the DV_ACCTMGR user. This way, if this user forgets his or her password, you can log in as the original DV_ACCTMGR account and reset the user's password. Otherwise, you must disable Oracle Database Vault, log in as SYS or SYSTEM to recreate the password, and then re-enable Database Vault.

How Are GRANT and REVOKE Operations Affected by DV_ACCTMGR?

Any account, such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone does not have the rights to grant this role to or revoke this role from any other database account.

The account with the DV_ACCTMGR role and the ADMIN OPTION can grant this role to any given database account and revoke this role from another account.

DV_ACCTMGR Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_REALM_OWNER Database Vault Realm DBA Role

The DV_REALM_OWNER role is used for realm management.

Use the DV_REALM_OWNER role to manage database objects in multiple schemas that define a realm.

Grant this role to the database account who is responsible for managing one or more schema database accounts within a realm and the roles associated with the realm.

Privileges Associated with the DV_REALM_OWNER Role

A user who has been granted this role can use powerful system privileges like CREATE ANY, ALTER ANY, and DROP ANY within the realm.

However, before this user can exercise these privileges, you must make this user either a participant or an owner for the realm. See About Realm Authorization for instructions.

There are no object privileges granted to the DV_REALM_OWNER role, but it does have some system privileges. To find the full list of DV_REALM_OWNER system privileges, log into the database instance with sufficient privileges and enter the following query:

```
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_REALM_OWNER';
```

How Are GRANT and REVOKE Operations Affected by DV_REALM_OWNER?

The realm owner of the Oracle System Privilege and Role Management realm, such as SYS, can grant this role to any given database account or role.

Note that though this role has powerful system privileges, it does not have any Oracle Database Vault roles such as the DV_OWNER or DV_ADMIN roles.

If you want to attach this role to a specific realm, then you must assign it to an account or business-related role, then authorize that account or role in the realm.

DV_REALM_OWNER Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_REALM_RESOURCE Database Vault Application Resource Owner Role

The DV_REALM_RESOURCE role is use for the management of realm resources.

Use the DV_REALM_RESOURCE role for operations such as creating tables, views, triggers, synonyms, and other objects that a realm would typically use.

Privileges Associated with the DV_REALM_RESOURCE Role

The DV_REALM_RESOURCE role provides the same system privileges as the Oracle RESOURCE role. In addition, both CREATE SYNONYM and CREATE VIEW are granted to this role.

There are no object privileges granted to the DV_REALM_RESOURCE role, but it does have some system privileges. To find the full list of DV_REALM_RESOURCE system privileges, log into the database instance with sufficient privileges and enter the following query:

```
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_REALM_RESOURCE';
```

Though this role has powerful system privileges, it does not have any Oracle Database Vault roles such as the DV_OWNER or DV_ADMIN roles.

How Are GRANT and REVOKE Operations Affected by DV_REALM_RESOURCE?

You can grant the DV_REALM_RESOURCE role to a database account that owns database tables, objects, triggers, views, procedures, and so on that are used to support any database application.

This is a role designed for a schema type database account. The realm owner of the Oracle System Privilege and Role Management realm, such as SYS, can grant this role to any database account or role.

DV_REALM_RESOURCE Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_POLICY_OWNER Database Vault Owner Role

The DV_POLICY_OWNER role enables database users to manage to a limited degree Oracle Database Vault policies.

Privileges Associated with the DV_POLICY_OWNER Role

The DV_POLICY_OWNER role provides non-Database Vault administrative users the sufficient privileges to enable or disable a Database Vault policy, add or remove authorization to or from a realm, and use the SELECT privilege for the following database views:

- DVSYS.POLICY_OWNER_COMMAND_RULE
- DVSYS.POLICY_OWNER_POLICY
- DVSYS.POLICY_OWNER_REALM
- DVSYS.POLICY_OWNER_REALM_AUTH
- DVSYS.POLICY_OWNER_REALM_OBJECT
- DVSYS.POLICY_OWNER_RULE_SET
- DVSYS.POLICY_OWNER_RULE
- DVSYS.POLICY_OWNER_RULE_SET_RULE

Only the DV_POLICY_OWNER can query these views. Even users who have the DV_OWNER and DV_ADMIN roles cannot query these views.

The DV_POLICY_OWNER role does not have any system privileges. To find the full list of object privileges that are associated with the DV_POLICY_OWNER role, you can log into the database instance enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE =
'DV_POLICY_OWNER';
```

How Are GRANT and REVOKE Operations Affected by DV_POLICY_OWNER?

Users who have been granted `DV_POLICY_OWNER` role cannot grant or revoke this role to or from other users.

DV_POLICY_OWNER Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the `GRANT ANY ROLE` system privilege can perform `GRANT` and `REVOKE` operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DV_PUBLIC Database Vault PUBLIC Role

The `DV_PUBLIC` role is no longer used.

The `DV_PUBLIC` role is still created during installation, but it is not granted any roles or privileges. All privileges that were granted to `DV_PUBLIC` in previous releases are now granted directly to the `PUBLIC` role.

Oracle Database Vault Accounts Created During Registration

You must create accounts for the Oracle Database Vault Owner and Oracle Database Vault Account Manager during the registration process.

You must supply an account name and password for the Oracle Database Vault Owner accounts during installation. Creating an Oracle Database Vault Account Manager is optional but strongly recommended for better separation of duty.

The Oracle Database Vault Owner account is granted the `DV_OWNER` role. This account can manage Oracle Database Vault roles and configuration.

The Oracle Database Vault Account Manager account is granted the `DV_ACCTMGR` role. This account is used to manage database user accounts to facilitate separation of duties.

If you choose not to create the Oracle Database Vault Account Manager account during installation, then both the `DV_OWNER` and `DV_ACCTMGR` roles are granted to the Oracle Database Vault Owner user account.

Table 14-2 lists the Oracle Database Vault database accounts that are needed in addition to the accounts that you create during installation.

Table 14-2 Database Accounts Used by Oracle Database Vault

Database Account	Roles and Privileges	Description
DVSYS	Several system and object privileges are provided to support Oracle Database Vault. The ability to create a session with this account is revoked at the end of the installation, and the account is locked.	Owner of Oracle Database Vault schema and related objects
DVF	A limited set of system privileges are provided to support Oracle Database Vault. The ability to create a session with this account is revoked at the end of the installation, and the account is locked.	Owner of the Oracle Database Vault functions that are created to retrieve factor identities
LBACSYS	This account is created when you install Oracle Label Security by using the Oracle Universal Installer custom installation option. (It is not created when you install Oracle Database Vault.) Do not drop or re-create this account. If you plan to integrate a factor with an Oracle Label Security policy, you must assign this user as the owner of the realm that uses this factor. See Using Oracle Database Vault Factors with Oracle Label Security Policies for more information.	Owner of the Oracle Label Security schema

You can create different database accounts to implement the separation of duties requirements for Oracle Database Vault. Table 14-3 lists some model database accounts that can act as a guide. (The accounts listed in Table 14-3 serve as a guide to implementing Oracle Database Vault roles. These are not actual accounts that are created during installation.)

Table 14-3 Model Oracle Database Vault Database Accounts

Database Account	Roles and Privileges	Description
EBROWN	DV_OWNER (with DV_ADMIN and DV_SECANALYST)	Account that is the realm owner for the Oracle Database Vault realm. This account can: • Execute DVSYS packages • Grant privileges on the DVSYS schema objects • Select objects in the DVSYS schema • Monitor Oracle Database Vault activity • Run reports on the Oracle Database Vault configuration

Table 14-3 (Cont.) Model Oracle Database Vault Database Accounts

Database Account	Roles and Privileges	Description
JGODFREY	DV_ACCTMGR	Account for administration of database accounts and profiles. This account can: • Create, alter, and drop users • Create, alter, and drop profiles • Grant and revoke the CREATE SESSION privilege • Grant and revoke the DV_ACCTMGR role, but only if this account was created during the Database Vault installation (this account is created with the ADMIN option) • Grant and revoke the CONNECT role **Note:** This account cannot create roles, or grant the RESOURCE or DBA roles.
RLAYTON	DV_ADMIN (with DV_SECANALYST)	Account to serve as the access control administrator. This account can: • Execute DVSYS packages • Monitor Oracle Database Vault activity • Run reports on the Oracle Database Vault configuration **Note:** This account cannot directly update the DVSYS tables.
PSMYTHE	DV_SECANALYST	Account for running Oracle Database Vault reports

Related Topics

- Configuring Oracle Database Vault Accounts as Enterprise User Accounts
  You can configure existing Oracle Database Vault user accounts as enterprise user accounts.
- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the DV_OWNER and DV_ACCTMGR roles.

Backup Oracle Database Vault Accounts

As a best practice, you should maintain backup accounts for the DV_OWNER and DV_ACCTMGR roles.

The Oracle Database Vault registration process entails creating both day-to-day and backup accounts for the DV_OWNER and DV_ACCTMGR roles. You should keep and maintain these accounts as a safety measure in case a user who has been granted one of these roles forgets his or her password or leaves the organization. Then you can log in to the backup account to recover the password or grant the role to a new account. These should be only used as a backup account kept safe in a privileged account management system or an organization break-glass (or emergency password recovery) system. When you grant a user one of these roles, include the WITH ADMIN OPTION clause in the GRANT statement.

Because of the strong separation of duty that Oracle Database Vault implements, loss of access to the DV_OWNER account will force you to rebuild the database. The SYS account cannot override the DV_OWNER account

Related Topics

- Resetting Oracle Database Vault Account Passwords
  Backup accounts can help you reset lost passwords for users who have been granted the DV_OWNER and DV_ACCTMGR roles.

15

Oracle Database Vault Realm APIs

The DBMS_MACADM PL/SQL package enables you to configure Oracle Database Vault realms.

Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures. For constants that you can use with these procedures, see Table 21-1 for more information.

- ADD_AUTH_TO_REALM Procedure
  The ADD_AUTH_TO_REALM procedure authorizes a user or role to access a realm as an owner or a participant. In a multitenant environment, you can authenticate both common and local realms.
- ADD_OBJECT_TO_REALM Procedure
  The ADD_OBJECT_TO_REALM procedure registers a set of objects for realm protection.
- CREATE_REALM Procedure
  The CREATE_REALM procedure creates a realm. In a multitenant environment, you can create both common and local realms.
- DELETE_AUTH_FROM_REALM Procedure
  The DELETE_AUTH_FROM_REALM procedure removes the authorization of a user or role to access a realm.
- DELETE_OBJECT_FROM_REALM Procedure
  The DELETE_OBJECT_FROM_REALM procedure removes a set of objects from realm protection.
- DELETE_REALM Procedure
  The DELETE_REALM procedure deletes a realm, including its related configuration information that specifies who is authorized and what objects are protected.
- DELETE_REALM_CASCADE Procedure
  The DELETE_REALM_CASCADE procedure deletes a realm, including its related Database Vault configuration information that specifies who is authorized and the objects that are protected.
- RENAME_REALM Procedure
  The RENAME_REALM procedure renames a realm; the name change takes effect everywhere the realm is used.
- UPDATE_REALM Procedure
  The UPDATE_REALM procedure updates a realm.
- UPDATE_REALM_AUTH Procedure
  The UPDATE_REALM_AUTH procedure updates the authorization of a user or role to access a realm.

ADD_AUTH_TO_REALM Procedure

The ADD_AUTH_TO_REALM procedure authorizes a user or role to access a realm as an owner or a participant. In a multitenant environment, you can authenticate both common and local realms.

For detailed information about realm authorization, see About Realm Authorization.

Optionally, you can specify a rule set that must be checked before allowing the authorization to be enabled.

Syntax

```
DBMS_MACADM.ADD_AUTH_TO_REALM(
 realm_name    IN VARCHAR2,
 grantee       IN VARCHAR2,
 rule_set_name IN VARCHAR2,
 auth_options  IN NUMBER
 auth_scope    IN NUMBER DEFAULT);
```

Parameters

Table 15-1 ADD_AUTH_TO_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.
grantee	User or role name to authorize as an owner or a participant. To find the existing users and roles in the current database instance, query the DBA_USERS and DBA_ROLES views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view, described in DBA_DV_REALM_AUTH View. To find existing secure application roles used in privilege management, query the DBA_DV_ROLE view. Both are described in Oracle Database Vault Data Dictionary Views.
rule_set_name	Optional. The rule set to check during runtime. The realm authorization is enabled only if the rule set evaluates to TRUE. To find the available rule sets, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET_RULE View.

Table 15-1 (Cont.) ADD_AUTH_TO_REALM Parameters

Parameter	Description
auth_options	Optional. Specify one of the following options to authorize the realm: • DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT: Participant. This account or role provides system or direct privileges to access, manipulate, and create objects protected by the realm, provided these rights have been granted using the standard Oracle Database privilege grant process. (Default) • DBMS_MACUTL.G_REALM_AUTH_OWNER: Owner. This account or role has the same authorization as the realm participant, plus the authorization to grant or revoke realm-secured roles and privileges on realm-protected objects. See About Realm Authorization for more information on participants and owners.
auth_scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) to authorize the realm locally in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) to authorize the realm in the application root

Examples

The following example authorizes user SYSADM as a participant in the Performance Statistics Realm. Because the default is to authorize the user as a participant, the auth_options parameter can be omitted.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name  => 'Performance Statistics Realm',
  grantee     => 'SYSADM');
END;
/
```

This example sets user SYSADM as the owner of the Performance Statistics Realm.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name   => 'Performance Statistics Realm',
  grantee      => 'SYSADM',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

The next example triggers the Check Conf Access rule set before allowing user SYSADM to act as the owner of the Performance Statistics Realm.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    => 'Performance Statistics Realm',
  grantee       => 'SYSADM',
  rule_set_name => 'Check Conf Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

This example shows how to commonly grant the common user C##HR_ADMIN access to the common realm HR Statistics Realm. The user running this procedure must be in the CDB root, and the rule set must be a common rule set residing in the application root.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    => 'HR Statistics Realm',
  grantee       => 'C##HR_ADMIN',
  rule_set_name => 'Check Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER,
 auth_scope    => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This example shows how to locally grant the common user C##HR_CLERK access to the common realm HR Statistics Realm. The user running this procedure must be in the same PDB in which the authorization applies. To find the existing PDBs query the DBA_PDBS data dictionary view. The rule set must be a local rule set.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    => 'HR Statistics Realm',
  grantee       => 'C##HR_CLERK',
  rule_set_name => 'Check Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER,
  auth_scope    => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

ADD_OBJECT_TO_REALM Procedure

The ADD_OBJECT_TO_REALM procedure registers a set of objects for realm protection.

Syntax

```
DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   IN VARCHAR2,
  object_owner IN VARCHAR2,
  object_name  IN VARCHAR2,
  object_type  IN VARCHAR2);
```

Parameters

Table 15-2 ADD_OBJECT_TO_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.

Table 15-2 (Cont.) ADD_OBJECT_TO_REALM Parameters

Parameter	Description
object_owner	The owner of the object that is being added to the realm. If you add a role to a realm, the object owner of the role is shown as % (for all), because roles do not have owners. To find the available users, query the DBA_USERS view, described in *Oracle Database Reference*. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view, described in DBA_DV_REALM_AUTH View.
object_name	Object name. (The wildcard % is allowed. See "Object Name" under About Realm-Secured Objects for exceptions to the wildcard %.) You can also use the DBMS_MACUTL.G_ALL_OBJECT constant. To find the available objects, query the ALL_OBJECTS view, described in *Oracle Database Reference*. To find objects that are secured by existing realms, query the DBA_DV_REALM_OBJECT view, described in DBA_DV_REALM_OBJECT View.
object_type	Object type, such as TABLE, INDEX, or ROLE. (The wildcard % is allowed. See "Object Types" under About Realm-Secured Objects for exceptions to the wildcard %.) You can also use the DBMS_MACUTL.G_ALL_OBJECT constant.

Example

```
BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Performance Statistics Realm',
  object_owner => '%',
  object_name  => 'GATHER_SYSTEM_STATISTICS',
  object_type  => 'ROLE');
END;
/
```

CREATE_REALM Procedure

The CREATE_REALM procedure creates a realm. In a multitenant environment, you can create both common and local realms.

After you create the realm, use the following procedures to complete the realm definition:

- ADD_OBJECT_TO_REALM procedure registers one or more objects for the realm.
- ADD_AUTH_TO_REALM procedure authorizes users or roles for the realm.

Syntax

```
DBMS_MACADM.CREATE_REALM(
 realm_name    IN VARCHAR2,
 description   IN VARCHAR2,
 enabled       IN VARCHAR2,
 audit_options IN NUMBER,
```

```
realm_type    IN NUMBER DEFAULT,
realm_scope   IN NUMBER DEFAULT
pl_sql_stack  IN BOOLEAN DEFAULT);
```

Parameters

Table 15-3 CREATE_REALM Parameters

Parameter	Description
realm_name	Realm name, up to 128 characters in mixed-case. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.
description	Description of the purpose of the realm, up to 1024 characters in mixed-case.
enabled	Specify one of the following options to set the status of the realm: • DBMS_MACUTL.G_YES or 'y' to enable realm checking (default) • DBMS_MACUTL.G_NO or 'n' to disable all realm checking, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
audit_options	Specify one of the following options to audit the realm: • DBMS_MACUTL.G_REALM_AUDIT_OFF: Disables auditing for the realm (default) • DBMS_MACUTL.G_REALM_AUDIT_FAIL: Creates an audit record when a realm violation occurs (for example, when an unauthorized user tries to modify an object that is protected by the realm) • DBMS_MACUTL.G_REALM_AUDIT_SUCCESS: Creates an audit record for authorized activities on objects protected by the realm • DBMS_MACUTL.G_REALM_AUDIT_FAIL + DBMS_MACUTL.G_REALM_AUDIT_SUCCESS: Creates an audit record for both authorized and unauthorized activities on objects protected by the realm
realm_type	Specify one of the following options: • 0: Disables mandatory realm checking. • 1: Enables mandatory realm checking for realm objects. Only realm owners or realm participants will have access to objects in a realm. Object owners and object-privileged users who are not realm owners or participants will have no access. See also Mandatory Realms to Restrict User Access to Objects within a Realm for more information about mandatory realms.
realm_scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the realm must be local in the current PDB. • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the realm must be in the application root. This setting duplicates the realm in all of the associated PDBs. If you create the common realm in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`

Table 15-3 (Cont.) CREATE_REALM Parameters

Parameter	Description
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record. The default is FALSE.

Examples

The following example shows how to create a realm that is enabled, has auditing set to track both failed and successful access, uses mandatory realm checking, and records the PL/SQL stack.

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Performance Statistics Realm',
  description   => 'Realm to measure performance',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL +
DBMS_MACUTL.G_REALM_AUDIT_SUCCESS,
  realm_type    => 1,
 pl_sql_stack  => TRUE);
END;
/
```

This example shows how to create a variation of the preceding example, but as a common realm located in the application root. The user who creates this realm must be a common user and must execute the procedure in the CDB root.

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Performance Statistics Realm',
  description   => 'Realm to measure performance',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL +
DBMS_MACUTL.G_REALM_AUDIT_SUCCESS,
  realm_type    => 1,
 realm_scope   => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This example shows how to create a local version n of the preceding example. The user who creates this realm must be in the PDB in which the realm will reside. To find existing PDBs, query the DBA_PDBS data dictionary view.

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Performance Statistics Realm',
  description   => 'Realm to measure performance',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL +
DBMS_MACUTL.G_REALM_AUDIT_SUCCESS,
  realm_type    => 1,
  realm_scope   => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

> **See Also:**
>
> Example 21-1

DELETE_AUTH_FROM_REALM Procedure

The DELETE_AUTH_FROM_REALM procedure removes the authorization of a user or role to access a realm.

Syntax

```
DBMS_MACADM.DELETE_AUTH_FROM_REALM(
 realm_name    IN VARCHAR2,
 grantee       IN VARCHAR2,
 auth_scope    IN NUMBER DEFAULT);
```

Parameters

Table 15-4 DELETE_AUTH_FROM_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View
grantee	User or role name. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view, described in DBA_DV_REALM_AUTH View.
auth_scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the realm was authorized locally in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2 if the realm was authorized in the application root

Example

```
BEGIN
DBMS_MACADM.DELETE_AUTH_FROM_REALM(
 realm_name  => 'Performance Statistics Realm',
 grantee     => 'PSMITH',
 auth_scope  => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

DELETE_OBJECT_FROM_REALM Procedure

The DELETE_OBJECT_FROM_REALM procedure removes a set of objects from realm protection.

Syntax

```
DBMS_MACADM.DELETE_OBJECT_FROM_REALM(
  realm_name   IN VARCHAR2,
  object_owner IN VARCHAR2,
  object_name  IN VARCHAR2,
  object_type  IN VARCHAR2);
```

Parameters

Table 15-5 DELETE_OBJECT_FROM_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.
object_owner	The owner of the object that was added to the realm. To find the available users, query the DBA_USERS view, described in *Oracle Database Reference*.
object_name	Object name. (The wildcard % is allowed. See "Object Name" under About Realm-Secured Objects for exceptions to the wildcard %.) You can also use the DBMS_MACUTL.G_ALL_OBJECT constant. To find objects that are secured by existing realms, query the DBA_DV_REALM_OBJECT view, described in DBA_DV_REALM_OBJECT View.
object_type	Object type, such as TABLE, INDEX, or ROLE. (The wildcard % is allowed. See "Object Types" under About Realm-Secured Objects for exceptions to the wildcard %.) You can also use the DBMS_MACUTL.G_ALL_OBJECT constant.

Example

```
BEGIN
 DBMS_MACADM.DELETE_OBJECT_FROM_REALM(
  realm_name   => 'Performance Statistics Realm',
  object_owner => 'SYS',
  object_name  => 'GATHER_SYSTEM_STATISTICS',
  object_type  => 'ROLE');
END;
/
```

DELETE_REALM Procedure

The DELETE_REALM procedure deletes a realm, including its related configuration information that specifies who is authorized and what objects are protected.

This procedure does not delete the actual database objects or users.

To find users who are authorized for the realm, query the DBA_DV_REALM_AUTH view. To find the objects that are protected by the realm, query the DBA_DV_REALM_OBJECT view. These views are described in Oracle Database Vault Data Dictionary Views.

Syntax

```
DBMS_MACADM.DELETE_REALM(
  realm_name IN VARCHAR2);
```

Parameters

Table 15-6 DELETE_REALM Parameter

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.

Example

```
EXEC DBMS_MACADM.DELETE_REALM('Performance Statistics Realm');
```

DELETE_REALM_CASCADE Procedure

The DELETE_REALM_CASCADE procedure deletes a realm, including its related Database Vault configuration information that specifies who is authorized and the objects that are protected.

The DBA_DV_REALM_AUTH view lists who is authorized in the realm and the DBA_DV_REALM_OBJECT view lists the protected objects.

It does not delete the actual database objects or users. This procedure works the same as the DELETE_REALM procedure. (In previous releases, these procedures were different, but now they are the same. Both are retained for earlier compatibility.) To find a listing of the realm-related objects, query the DBA_DV_REALM view. To find its authorizations, query DBA_DV_REALM_AUTH. Both are described under Oracle Database Vault Data Dictionary Views.

Syntax

```
DBMS_MACADM.DELETE_REALM_CASCADE(
  realm_name IN VARCHAR2);
```

Parameters

Table 15-7 DELETE_REALM_CASCADE Parameter

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.

Example

```
EXEC DBMS_MACADM.DELETE_REALM_CASCADE('Performance Statistics Realm');
```

RENAME_REALM Procedure

The RENAME_REALM procedure renames a realm; the name change takes effect everywhere the realm is used.

Syntax

```
DBMS_MACADM.RENAME_REALM(
 realm_name  IN VARCHAR2,
 new_name    IN VARCHAR2);
```

Parameters

Table 15-8 RENAME_REALM Parameters

Parameter	Description
realm_name	Current realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.
new_name	New realm name, up to 128 characters in mixed-case.

Example

```
BEGIN
 DBMS_MACADM.RENAME_REALM(
  realm_name => 'Performance Statistics Realm',
  new_name   => 'Sector 2 Performance Statistics Realm');
END;
/
```

UPDATE_REALM Procedure

The UPDATE_REALM procedure updates a realm.

To find information about the current settings for a realm, query the DVSYS.DV$REALM view, described in DVSYS.DV$REALM View.

Syntax

```
DBMS_MACADM.UPDATE_REALM(
 realm_name    IN VARCHAR2,
 description   IN VARCHAR2,
 enabled       IN VARCHAR2,
 audit_options IN NUMBER DEFAULT NULL,
 realm_type    IN NUMBER DEFAULT NULL
 pl_sql_stack  IN BOOLEAN DEFAULT NULL);
```

Parameters

Table 15-9 UPDATE_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.
description	Description of the purpose of the realm, up to 1024 characters in mixed-case.
enabled	Specify one of the following options to set the status of the realm: • DBMS_MACUTL.G_YES or 'y' to enable realm checking • DBMS_MACUTL.G_NO or 'n' to disable all realm checking, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log The default for enabled is the previously set value, which you can find by querying the DBA_DV_REALM data dictionary view.
audit_options	Specify one of the following options to audit the realm: • DBMS_MACUTL.G_REALM_AUDIT_OFF: Disables auditing for the realm • DBMS_MACUTL.G_REALM_AUDIT_FAIL: Creates an audit record when a realm violation occurs (for example, when an unauthorized user tries to modify an object that is protected by the realm • DBMS_MACUTL.G_REALM_AUDIT_SUCCESS: Creates an audit record for authorized activities on objects protected by the realm. • DBMS_MACUTL.G_REALM_AUDIT_FAIL + DBMS_MACUTL.G_REALM_AUDIT_SUCCESS: Creates an audit record for both authorized and unauthorized activities on objects protected by the realm The default for audit_options is the previously set value, which you can find by querying the DBA_DV_REALM data dictionary view.
realm_type	If you do not specify the realm_type parameter, then Oracle Database Vault does not update the current realm_type setting. Specify one of the following options: • 0: Sets the realm to be a regular realm, which does not have mandatory realm checking. • 1: Enables mandatory realm checking for realm objects. Only realm owners or realm participants will have access to objects in a realm. Object owners and object-privileged users who are not realm owners or participants will have no access. See also Mandatory Realms to Restrict User Access to Objects within a Realm for more information about mandatory realms.
pl_sql_stack	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_REALM(
  realm_name    => 'Sector 2 Performance Statistics Realm',
  description   => 'Realm to measure performance for Sector 2 applications',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL +
DBMS_MACUTL.G_REALM_AUDIT_SUCCESS),
  realm_type    => 1);
END;
/
```

UPDATE_REALM_AUTH Procedure

The UPDATE_REALM_AUTH procedure updates the authorization of a user or role to access a realm.

Syntax

```
DBMS_MACADM.UPDATE_REALM_AUTH(
 realm_name    IN VARCHAR2,
 grantee       IN VARCHAR2,
 rule_set_name IN VARCHAR2,
 auth_options  IN NUMBER,
 auth_scope    IN NUMBER DEFAULT);
```

Parameters

Table 15-10 UPDATE_REALM_AUTH Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view, described in DBA_DV_REALM View.
grantee	User or role name. To find the available users and roles in the current database instance, query the DBA_USERS and DBA_ROLES views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view, described in DBA_DV_REALM_AUTH View. To find existing secure application roles used in privilege management, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View.
rule_set_name	Optional. A rule set to check during runtime. The realm authorization is enabled only if the rule set evaluates to TRUE. To find the available rule sets, query the DBA_DV_RULE_SET view. To find rules that are associated with the rule sets, query the DBA_DB_RULE_SET_RULE view. Both are described in Oracle Database Vault Data Dictionary Views.

Table 15-10 (Cont.) UPDATE_REALM_AUTH Parameters

Parameter	Description
auth_options	Optional. Specify one of the following options to authorize the realm: • DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT: Participant. This account or role provides system or direct privileges to access, manipulate, and create objects protected by the realm, provided these rights have been granted using the standard Oracle Database privilege grant process. • DBMS_MACUTL.G_REALM_AUTH_OWNER: Owner. This account or role has the same authorization as the realm participant, plus the authorization to grant or revoke realm-secured roles and privileges on realm-protected objects. A realm can have multiple owners. The default for auth_options value is the previously set value, which you can find by querying the DBA_DV_REALM_AUTH data dictionary view.
realm_auth	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the realm is authorized locally in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the realm is authorized in the application root

Example

```
BEGIN
 DBMS_MACADM.UPDATE_REALM_AUTH(
  realm_name    => 'Sector 2 Performance Statistics Realm',
  grantee       => 'SYSADM',
  rule_set_name => 'Check Conf Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

16

Oracle Database Vault Rule Set APIs

You can use the DBMS_MACADM PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.

- DBMS_MACADM Rule Set Procedures
  The DBMS_MACADM rule set procedures enable you to configure both rule sets and individual rules that go within these rule sets.
- Oracle Database Vault PL/SQL Rule Set Functions
  Oracle Database Vault provides functions to use in rule sets to inspect the SQL statement that the rule set protects.

DBMS_MACADM Rule Set Procedures

The DBMS_MACADM rule set procedures enable you to configure both rule sets and individual rules that go within these rule sets.

Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures.

- ADD_RULE_TO_RULE_SET Procedure
  The ADD_RULE_TO_RULE_SET procedure adds rule to a rule set; you can enable having the rule checked when the rule set is evaluated.
- CREATE_RULE Procedure
  The CREATE_RULE procedure creates a rule, which afterwards, can be added to a rule set.
- CREATE_RULE_SET Procedure
  The CREATE_RULE_SET procedure creates a rule set.
- DELETE_RULE Procedure
  The DELETE_RULE procedure deletes a rule.
- DELETE_RULE_FROM_RULE_SET Procedure
  The DELETE_RULE_FROM_RULE_SET procedure deletes a rule from a rule set.
- DELETE_RULE_SET Procedure
  The DELETE_RULE_SET procedure deletes a rule set.
- RENAME_RULE Procedure
  The RENAME_RULE procedure renames a rule and causes the name change to take effect everywhere the rule is used
- RENAME_RULE_SET Procedure
  The RENAME_RULE_SET procedure renames a rule set and causes the name change to take effect everywhere the rule set is used.
- UPDATE_RULE Procedure
  The UPDATE_RULE procedure updates a rule.
- UPDATE_RULE_SET Procedure
  The UPDATE_RULE_SET procedure updates a rule set.

Related Topics

- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the DBMS_MACUTL PL/SQL package.

ADD_RULE_TO_RULE_SET Procedure

The ADD_RULE_TO_RULE_SET procedure adds rule to a rule set; you can enable having the rule checked when the rule set is evaluated.

Syntax

```
DBMS_MACADM.ADD_RULE_TO_RULE_SET(
 rule_set_name  IN VARCHAR2,
 rule_name      IN VARCHAR2,
 rule_order     IN NUMBER,
 enabled        IN VARCHAR2,
 scope          IN NUMBER DEFAULT);
```

Parameters

Table 16-1 ADD_RULE_TO_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
rule_name	Rule to add to the rule set. To find existing rules, query the DBA_DV_RULE view, described in DBA_DV_RULE View. To find rules that have been associated with rule sets, use DBA_DV_RULE_SET_RULE, described in DBA_DV_RULE View.
rule_order	Does not apply to this release, but you must include a value for the ADD_RULE_TO_RULE_SET procedure to work. Enter 1.
enabled	Optional. Determines whether the rule should be checked when the rule set is evaluated. Possible values are: • DBMS_MACUTL.G_YES (default). Enables the rule to be checked during the rule set evaluation. • DBMS_MACUTL.G_NO Prevents the rule from being checked during the rule set evaluation. See Table 21-1 for more information.
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule and rule set are local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule and rule set are in the application root

Examples

The following example adds a rule to a rule set, and by omitting the `enabled` parameter, automatically enables the rule to be checked when the rule set is evaluated.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Restrict DROP TABLE operations',
  rule_order    => 1);
END;
/
```

This example adds the rule to the rule set but disables rule checking.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Check UPDATE operations',
  rule_order    => 1,
  enabled       => DBMS_MACUTL.G_NO);
END;
/
```

CREATE_RULE Procedure

The `CREATE_RULE` procedure creates a rule, which afterwards, can be added to a rule set.

In a multitenant environment, you can create both common and local rules.

Syntax

```
DBMS_MACADM.CREATE_RULE(
 rule_name  IN VARCHAR2,
 rule_expr  IN VARCHAR2
 scope      IN NUMBER DEFAULT);
```

Parameters

Table 16-2 CREATE_RULE Parameters

Parameter	Description
`rule_name`	Rule name, up to 128 characters in mixed-case. Spaces are allowed. To find existing rules in the current database instance, query the `DBA_DV_RULE` view, described in DBA_DV_RULE View. To find rules that have been associated with rule sets, query `DBA_DV_RULE_SET_RULE`, described in DBA_DV_RULE_SET_RULE View.

Table 16-2 (Cont.) CREATE_RULE Parameters

Parameter	Description
rule_expr	PL/SQL BOOLEAN expression. If the expression contains quotation marks, do not use double quotation marks. Instead, use two single quotation marks. Enclose the entire expression within single quotation marks. For example: `'TO_CHAR(SYSDATE,''HH24'') = ''12'''` See Creating a New Rule for more information on rule expressions.
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule is in the application root

Examples

The following example shows how to create a local rule expression that checks if the current session user is SYSADM. The user running this procedure must be in the same PDB in which the rule and its rule set reside. To find the existing PDBs, run the show pdbs command. The rule and rule set must be local.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Check UPDATE operations',
  rule_expr  =>'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SYSADM''',
  scope      => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

This example shows a multitenant environment common version of the preceding example. The user running this procedure must be in the CDB root, and the rule and its associated rule set must be common. The rule will reside in the application root.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Check UPDATE operations',
  rule_expr  =>'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SYSADM''',
  scope      => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This example shows how to create a rule expression that uses the public standalone function OLS_LABEL_DOMINATES to find if the session label of the hr_ols_pol Oracle Label Security policy dominates or is equal to the hs label. The value 0 indicates if it is false. (To check if it is equal, you would specify 1.)

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Check OLS Factor',
  rule_expr  => 'OLS_LABEL_DOMINATES(''hr_ols_pol'', ''hs'') = 1');
END;
/
```

CREATE_RULE_SET Procedure

The CREATE_RULE_SET procedure creates a rule set.

After you create a rule set, you can use the CREATE_RULE and ADD_RULE_TO_RULE_SET procedures to create and add rules to the rule set.

Syntax

```
DBMS_MACADM.CREATE_RULE_SET(
 rule_set_name    IN VARCHAR2,
 description      IN VARCHAR2,
 enabled          IN VARCHAR2,
 eval_options     IN NUMBER,
 audit_options    IN NUMBER,
 fail_options     IN NUMBER,
 fail_message     IN VARCHAR2,
 fail_code        IN NUMBER,
 handler_options  IN NUMBER,
 handler          IN VARCHAR2,
 is_static        IN BOOLEAN DEFAULT,
 scope            IN NUMBER DEFAULT);
```

Parameters

Table 16-3 CREATE_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name, up to 128 characters in mixed-case. Spaces are allowed. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
description	Description of the purpose of the rule set, up to 1024 characters in mixed-case.
enabled	DBMS_MACUTL.G_YES (Yes) enables the rule set; DBMS_MACUTL.G_NO (No) disables it. The default is DBMS_MACUTL.G_YES.
eval_options	If you plan to assign multiple rules to the rule set, enter one of the following settings: • DBMS_MACUTL.G_RULESET_EVAL_ALL: All rules in the rule set must evaluate to true for the rule set itself to evaluate to true (default). • DBMS_MACUTL.G_RULESET_EVAL_ANY: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to true.
audit_options	Select one of the following settings: • DBMS_MACUTL.G_RULESET_AUDIT_OFF: Disables auditing for the rule set (default) • DBMS_MACUTL.G_RULESET_AUDIT_FAIL: Creates an audit record when a rule set violation occurs • DBMS_MACUTL.G_RULESET_AUDIT_SUCCESS: Creates an audit record for a successful rule set evaluation • DBMS_MACUTL.G_RULESET_AUDIT_FAIL + DBMS_MACUTL.G_RULESET_AUDIT_SUCCESS: Creates an audit record for both successful and failed rule set evaluations

Table 16-3 (Cont.) CREATE_RULE_SET Parameters

Parameter	Description
fail_options	Options for reporting errors: • DBMS_MACUTL.G_RULESET_FAIL_SHOW: Shows an error message (default) • DBMS_MACUTL.G_RULESET_FAIL_SILENT: Does not show an error message
fail_message	Enter an error message for failure, up to 80 characters in mixed-case, to associate with the fail code you specify for fail_code.
fail_code	Enter a number in the range of -20000 to -20999 or 20000 to 20999 to associate with the fail_message parameter.
handler_options	Select one of the following settings: • DBMS_MACUTL.G_RULESET_HANDLER_OFF: Disables error handling (default) • DBMS_MACUTL.G_RULESET_HANDLER_FAIL: Calls handler on rule set failure • DBMS_MACUTL.G_RULESET_HANDLER_SUCCESS: Calls handler on rule set success
handler	Name of the PL/SQL function or procedure that defines the custom event handler logic.
is_static	Optional. Determines how often a rule set is evaluated when it is accessed. The default is FALSE. • TRUE: The rule set is evaluated once during the user session. After that, the value is re-used. • FALSE: The rule set is evaluated every time.
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule set is to be local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule set is to be in the application root

Examples

The following example creates a rule set that is enabled, is set so that at least one rule must evaluate to true for the rule set itself to evaluate to true, and audits both failed and successful attempts. It does not show error messages but uses the fail code 20461 to track failures. It also uses a handler to send email alerts to the appropriate users if their are violations to the rule set.

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL +
DBMS_MACUTL.G_RULESET_AUDIT_SUCCESS,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
  fail_message     => '',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
```

```
  handler          => 'dbavowner.email_alert',
  is_static        => TRUE);
END;
/
```

This rule set uses no fail messages or fail codes, nor does it use any handlers. This rule set will be in the application root of a multitenant environment, so the user running this procedure must be in the application root. Any rules or command rules that are associated with this rule set must be common.

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
 rule_set_name    => 'Check_HR_Access',
 description      => 'Checks for failed access attempts to the HR schema',
 enabled          => DBMS_MACUTL.G_YES,
 eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
 audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
 fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
 fail_message     => '',
 fail_code        => '',
 handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
 handler          => '',
 is_static        => TRUE,
 scope            => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This rule set is a local version of the preceding rule set. The user who creates this rule set must be in the PDB in which this rule set will reside. To find the existing PDBs, query the DBA_PDBS data dictionary view. Any rules or command rules that are associated with this rule set must be local.

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
 rule_set_name    => 'Check_HR_Access',
 description      => 'Checks for failed access attempts to the HR schema',
 enabled          => DBMS_MACUTL.G_YES,
 eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
 audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
 fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
 fail_message     => '',
 fail_code        => '',
 handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
 handler          => '',
 is_static        => TRUE,
 scope            => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

See Also:

Example 21-2

DELETE_RULE Procedure

The DELETE_RULE procedure deletes a rule.

Syntax

```
DBMS_MACADM.DELETE_RULE(
 rule_name IN VARCHAR2);
```

Parameter

Table 16-4 DELETE_RULE Parameter

Parameter	Description
rule_name	Rule name. To find existing rules in the current database instance, query the DBA_DV_RULE view, described in DBA_DV_RULE View. To find rules that have been associated with rule sets, query DBA_DV_RULE_SET_RULE, described in DBA_DV_RULE_SET_RULE View.

Example

```
EXEC DBMS_MACADM.DELETE_RULE('Check UPDATE operations');
```

DELETE_RULE_FROM_RULE_SET Procedure

The DELETE_RULE_FROM_RULE_SET procedure deletes a rule from a rule set.

Syntax

```
DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
 rule_set_name IN VARCHAR2,
 rule_name     IN VARCHAR2);
```

Parameters

Table 16-5 DELETE_RULE_FROM_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
rule_name	Rule to remove from the rule set. To find existing rules in the current database instance, query the DBA_DV_RULE view, described in DBA_DV_RULE View. To find rules that have been associated with rule sets, query DBA_DV_RULE_SET_RULE, described in DBA_DV_RULE_SET_RULE View.

Example

```
BEGIN
 DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Check UPDATE operations');
END;
/
```

DELETE_RULE_SET Procedure

The DELETE_RULE_SET procedure deletes a rule set.

Syntax

```
DBMS_MACADM.DELETE_RULE_SET(
 rule_set_name IN VARCHAR2);
```

Parameters

Table 16-6 DELETE_RULE_SET Parameter

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.

Example

```
EXEC DBMS_MACADM.DELETE_RULE_SET('Limit_DBA_Access');
```

RENAME_RULE Procedure

The RENAME_RULE procedure renames a rule and causes the name change to take effect everywhere the rule is used

Syntax

```
DBMS_MACADM.RENAME_RULE(
 rule_name  IN VARCHAR2,
 new_name   IN VARCHAR2,
 scope      IN NUMBER DEFAULT);
```

Parameters

Table 16-7 RENAME_RULE Parameters

Parameter	Description
rule_name	Current rule name. To find existing rules in the current database instance, query the DBA_DV_RULE view, described in DBA_DV_RULE View. To find rules that have been associated with rule sets, query DBA_DV_RULE_SET_RULE, described in DBA_DV_RULE_SET_RULE View.
new_name	New rule name, up to 128 characters in mixed-case.
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule is in the application root

Example

```
BEGIN
 DBMS_MACADM.RENAME_RULE(
  rule_name  => 'Check UPDATE operations',
  new_name   => 'Check Sector 2 Processes');
END;
/
```

RENAME_RULE_SET Procedure

The RENAME_RULE_SET procedure renames a rule set and causes the name change to take effect everywhere the rule set is used.

Syntax

```
DBMS_MACADM.RENAME_RULE_SET(
 rule_set_name IN VARCHAR2,
 new_name      IN VARCHAR2,
 scope         IN NUMBER DEFAULT);
```

Parameters

Table 16-8 RENAME_RULE_SET Parameters

Parameter	Description
rule_set_name	Current rule set name. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
new_name	New rule set name, up to 128 characters in mixed-case. Spaces are allowed.

Table 16-8 (Cont.) RENAME_RULE_SET Parameters

Parameter	Description
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule set is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule set is in the application root

Example

```
BEGIN
 DBMS_MACADM.RENAME_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  new_name      => 'Limit Sector 2 Access');
END;
/
```

UPDATE_RULE Procedure

The UPDATE_RULE procedure updates a rule.

Syntax

```
DBMS_MACADM.UPDATE_RULE(
 rule_name  IN VARCHAR2,
 rule_expr  IN VARCHAR2);
```

Parameters

Table 16-9 UPDATE_RULE Parameters

Parameter	Description
rule_name	Rule name. To find existing rules in the current database instance, query the DBA_DV_RULE view, described in DBA_DV_RULE View. To find rules that have been associated with rule sets, query DBA_DV_RULE_SET_RULE, described in DBA_DV_RULE_SET_RULE View.
rule_expr	PL/SQL BOOLEAN expression. If the expression contains quotation marks, do not use double quotation marks. Instead, use two single quotation marks. Enclose the entire expression within single quotation marks. For example: 'TO_CHAR(SYSDATE,''HH24'') = ''12''' See Creating a New Rule for more information on rule expressions. To find existing rule expressions, query the DBA_DV_RULE view.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_RULE(
  rule_name  => 'Check UPDATE operations',
  rule_expr  =>'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SYSADM'' AND
               (
                 UPPER(SYS_CONTEXT(''USERENV'',''MODULE'')) LIKE ''APPSRVR%'' OR
                 UPPER(SYS_CONTEXT(''USERENV'',''MODULE'')) LIKE ''DBAPP%'' )'
               );
END;
/
```

UPDATE_RULE_SET Procedure

The UPDATE_RULE_SET procedure updates a rule set.

Syntax

```
DBMS_MACADM.UPDATE_RULE_SET(
 rule_set_name    IN VARCHAR2,
 description      IN VARCHAR2,
 enabled          IN VARCHAR2,
 eval_options     IN NUMBER,
 audit_options    IN NUMBER,
 fail_options     IN NUMBER,
 fail_message     IN VARCHAR2,
 fail_code        IN NUMBER,
 handler_options  IN NUMBER,
 handler          IN VARCHAR2,
 is_static        IN BOOLEAN DEFAULT);
```

Parameters

Table 16-10 UPDATE_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
description	Description of the purpose of the rule set, up to 1024 characters in mixed-case.
enabled	DBMS_MACUTL.G_YES (Yes) enables rule set checking; DBMS_MACUTL.G_NO (No) disables it. The default for the enabled setting is the previously set value, which you can find by querying the DBA_DV_RULE_SET data dictionary view.
eval_options	If you plan to assign multiple rules to the rule set, enter one of the following settings: • DBMS_MACUTL.G_RULESET_EVAL_ALL: All rules in the rule set must evaluate to true for the rule set itself to evaluate to true. • DBMS_MACUTL.G_RULESET_EVAL_ANY: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to true. The default for eval_options is the previously set value, which you can find by querying the DBA_DV_RULE_SET data dictionary view.

Table 16-10 (Cont.) UPDATE_RULE_SET Parameters

Parameter	Description
audit_options	Select one of the following settings: • DBMS_MACUTL.G_RULESET_AUDIT_OFF: Disables auditing for the rule set • DBMS_MACUTL.G_RULESET_AUDIT_FAIL: Creates an audit record when a rule set violation occurs • DBMS_MACUTL.G_RULESET_AUDIT_SUCCESS: Creates an audit record for a successful rule set evaluation • DBMS_MACUTL.G_RULESET_AUDIT_FAIL + DBMS_MACUTL.G_RULESET_AUDIT_SUCCESS: Creates an audit record for both successful and failed rule set evaluations The default for audit_options is the previously set value, which you can find by querying the DBA_DV_RULE_SET data dictionary view.
fail_options	Options for reporting errors: • DBMS_MACUTL.G_RULESET_FAIL_SHOW: Shows an error message. • DBMS_MACUTL.G_RULESET_FAIL_SILENT: Does not show an error message. The default for fail_options is the previously set value, which you can find by querying the DBA_DV_RULE_SET data dictionary view.
fail_message	Error message for failure, up to 80 characters in mixed-case, to associate with the fail code you specify for fail_code.
fail_code	Enter a number in the range of -20000 to -20999 or 20000 to 20999 to associate with the fail_message parameter.
handler_options	Select one of the following settings: • DBMS_MACUTL.G_RULESET_HANDLER_OFF: Disables error handling. • DBMS_MACUTL.G_RULESET_HANDLER_FAIL: Call handler on rule set failure. • DBMS_MACUTL.G_RULESET_HANDLER_SUCCESS: Call handler on rule set success. The default for handler_options is the previously set value, which you can find by querying the DBA_DV_RULE_SET data dictionary view.
handler	Name of the PL/SQL function or procedure that defines the custom event handler logic.
is_static	Optional. Determines how often a rule set is evaluated when it is accessed by a SQL statement. The default is FALSE. • TRUE: The rule set is evaluated once during the user session. After that, the value is re-used. • FALSE: The rule set evaluated each time a SQL statement accesses it.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
```

```
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'Access denied!',
  fail_code        => 20900,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
  handler          => '',
  is_static        =  TRUE);
END;
/
```

Oracle Database Vault PL/SQL Rule Set Functions

Oracle Database Vault provides functions to use in rule sets to inspect the SQL statement that the rule set protects.

- DV_SYSEVENT Function
  The DV_SYSEVENT function returns the system event firing the rule set. .
- DV_LOGIN_USER Function
  The DV_LOGIN_USER function returns the login user name, in VARCHAR2 data type.
- DV_INSTANCE_NUM Function
  The DV_INSTANCE_NUM function returns the database instance number, in NUMBER data type.
- DV_DATABASE_NAME Function
  The DV_DATABASE_NAME function returns the database name, in VARCHAR2 data type.
- DV_DICT_OBJ_TYPE Function
  The DV_DICT_OBJ_TYPE function returns the type of the dictionary object on which the database operation occurred.
- DV_DICT_OBJ_OWNER Function
  The DV_DICT_OBJ_OWNER function returns the name of the owner of the dictionary object on which the database operation occurred.
- DV_DICT_OBJ_NAME Function
  The DV_DICT_OBJ_NAME function returns the name of the dictionary object on which the database operation occurred.
- DV_SQL_TEXT Function
  The DV_SQL_TEXT function returns the first 4000 characters of SQL text of the database statement used in the operation.

DV_SYSEVENT Function

The DV_SYSEVENT function returns the system event firing the rule set. .

The event name is the same as that in the syntax of the SQL statement (for example, INSERT, CREATE.) The return type is VARCHAR2.

Syntax

```
DV_SYSEVENT ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get System Event Firing the Maintenance Rule Set',
  rule_expr => 'DV_SYSEVENT = ''CREATE''');
END;
/
```

DV_LOGIN_USER Function

The DV_LOGIN_USER function returns the login user name, in VARCHAR2 data type.

Syntax

```
DV_LOGIN_USER ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check System Login User Name',
  rule_expr => 'DV_LOGIN_USER = ''SEBASTIAN''');
END;
/
```

DV_INSTANCE_NUM Function

The DV_INSTANCE_NUM function returns the database instance number, in NUMBER data type.

Syntax

```
DV_INSTANCE_NUM ()
RETURN NUMBER;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database Instance Number',
  rule_expr => 'DV_INSTANCE_NUM BETWEEN 6 AND 9');
END;
/
```

DV_DATABASE_NAME Function

The DV_DATABASE_NAME function returns the database name, in VARCHAR2 data type.

Syntax

```
DV_DATABASE_NAME ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database Name',
  rule_expr => 'DV_DATABASE_NAME = ''ORCL''');
END;
/
```

DV_DICT_OBJ_TYPE Function

The DV_DICT_OBJ_TYPE function returns the type of the dictionary object on which the database operation occurred.

For example, dictionary objects it returns are table, procedure, or view. The return type is VARCHAR2.

Syntax

```
DV_DICT_OBJ_TYPE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Dictionary Object Type',
  rule_expr => 'DV_DICT_OBJ_TYPE IN (''TABLE'', ''VIEW'')');
END;
/
```

DV_DICT_OBJ_OWNER Function

The DV_DICT_OBJ_OWNER function returns the name of the owner of the dictionary object on which the database operation occurred.

The return type is VARCHAR2.

Syntax

```
DV_DICT_OBJ_OWNER ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Dictionary Object Owner',
  rule_expr => 'DV_DICT_OBJ_OWNER = ''JSMITH''');
END;
/
```

DV_DICT_OBJ_NAME Function

The DV_DICT_OBJ_NAME function returns the name of the dictionary object on which the database operation occurred.

The return type is VARCHAR2.

Syntax

```
DV_DICT_OBJ_NAME ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Dictionary Object Name',
  rule_expr => 'DV_DICT_OBJ_NAME = ''SALES''');
END;
/
```

DV_SQL_TEXT Function

The DV_SQL_TEXT function returns the first 4000 characters of SQL text of the database statement used in the operation.

The return type is VARCHAR2.

Syntax

```
DV_SQL_TEXT ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check SQL Text',
  rule_expr => 'DV_SQL_TEXT = ''SELECT SALARY FROM HR.EMPLOYEES''');
END;
/
```

17

Oracle Database Vault Command Rule APIs

The DBMS_MACADM PL/SQL package provides procedures for configuring command rules. .

Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures.

- CREATE_COMMAND_RULE Procedure
  The CREATE_COMMAND_RULE procedure creates a command rule and associates it with a rule set.
- CREATE_CONNECT_COMMAND_RULE Procedure
  The CREATE_CONNECT_COMMAND_RULE procedure creates a CONNECT command rule that you can associate with a user and a rule set.
- CREATE_SESSION_EVENT_CMD_RULE Procedure
  The CREATE_SESSION_EVENT_CMD_RULE procedure creates a command rule that you can associate with session events, based on the ALTER SESSION statement.
- CREATE_SYSTEM_EVENT_CMD_RULE Procedure
  The CREATE_SYSTEM_EVENT_CMD_RULE procedure creates a command rule that you can associate with system events, based on the ALTER SYSTEM statement.
- DELETE_COMMAND_RULE Procedure
  The DELETE_COMMAND_RULE procedure drops a command rule declaration.
- DELETE_CONNECT_COMMAND_RULE Procedure
  The DELETE_CONNECT_COMMAND_RULE procedure deletes a CONNECT command rule that had been created with the CREATE_CONNECT_COMMAND_RULE procedure.
- DELETE_SESSION_EVENT_CMD_RULE Procedure
  The DELETE_SESSION_EVENT_CMD_RULE procedure deletes a session command rule that was associated with events.
- DELETE_SYSTEM_EVENT_CMD_RULE Procedure
  The DELETE_SYSTEM_EVENT_CMD_RULE procedure deletes a system command rule that was associated with events.
- UPDATE_COMMAND_RULE Procedure
  The UPDATE_COMMAND_RULE procedure updates a command rule declaration.
- UPDATE_CONNECT_COMMAND_RULE Procedure
  The UPDATE_CONNECT_COMMAND_RULE procedure updates a CONNECT command rule that had been created with the CREATE_CONNECT_COMMAND_RULE procedure.
- UPDATE_SESSION_EVENT_CMD_RULE Procedure
  The UPDATE_SESSION_EVENT_CMD_RULE procedure updates a session event command rule, based on the ALTER SESSION statement.
- UPDATE_SYSTEM_EVENT_CMD_RULE Procedure
  The UPDATE_SYSTEM_EVENT_CMD_RULE procedure updates a system event command rule, based on the ALTER SYSTEM statement.

Related Topics

- Configuring Command Rules
  You can create command rules or use the default command rules to protect DDL and DML statements.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

CREATE_COMMAND_RULE Procedure

The `CREATE_COMMAND_RULE` procedure creates a command rule and associates it with a rule set.

Optionally, you can use it to enable the command rule for rule checking with a rule set. In a multitenant environment, you can create both common and local command rules.

Syntax

```
DBMS_MACADM.CREATE_COMMAND_RULE(
 command         IN VARCHAR2,
 rule_set_name   IN VARCHAR2,
 object_owner    IN VARCHAR2,
 object_name     IN VARCHAR2,
 enabled         IN VARCHAR2,
 privilege_scope IN NUMBER,
 clause_name     IN VARCHAR2,
 parameter_name  IN VARCHAR2,
 event_name      IN VARCHAR2,
 component_name  IN VARCHAR2,
 action_name     IN VARCHAR2,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 17-1 CREATE_COMMAND_RULE Parameters

Parameter	Description
`command`	SQL statement to protect. See also the following: • Default Command Rules for information about default command rules • DBA_DV_COMMAND_RULE View for a listing of existing command rules • SQL Statements That Can Be Protected by Command Rules for a listing of available SQL statements that you can use
`rule_set_name`	Name of rule set to associate with this command rule. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view, described in DBA_DV_RULE_SET View.

Table 17-1 (Cont.) CREATE_COMMAND_RULE Parameters

Parameter	Description
object_owner	Database schema to which this command rule will apply. The wildcard % is allowed, except for the SELECT, INSERT, UPDATE, DELETE, and EXECUTE statements. To find the available users, query the DBA_USERS view, described in *Oracle Database Reference*. See also "Object Owner" in Creating a Command Rule for more information.
object_name	Object to be protected by the command rule. (The wildcard % is allowed. See "Object Name" in Creating a Command Rule for more information about objects protected by command rules.) To find the available objects, query the ALL_OBJECTS view, described in *Oracle Database Reference*.
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' (Yes) to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
privilege_scope	Obsolete parameter
clause_name	A clause from the SQL statement that was used to create the command rule. For example, a command rule for the ALTER SESSION SQL statement could have the SET clause as the clause_name parameter. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
parameter_name	A parameter from the clause_name parameter. For example, for an ALTER SESSION command rule, you could set parameter_name to EVENTS if the clause_name is SET. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
event_name	An event that the command rule defines. For example, suppose an ALTER SESSION command rule uses SET for the clause_name and EVENTS as the parameter_name. The event_name could be set to TRACE if you want to track trace events. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to EVENTS.
component_name	A component of the event_name setting. For example, for a TRACE event, the component_name could be GCS. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to EVENTS.
action_name	An action of the component_name setting. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to EVENTS.

Table 17-1 (Cont.) CREATE_COMMAND_RULE Parameters

Parameter	Description
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you create the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`

ALTER SYSTEM Command Rule Settings

Table 17-2 describes the ALTER SYSTEM command rule settings.

Table 17-2 ALTER SYSTEM Command Rule Settings

clause_name	parameter_name — Parameter Value
ARCHIVE LOG	• ALL — *sequence_number* • CHANGE — *change_number* • CURRENT — N/A • GROUP — *group_number* • LOGFILE — *log_file_name* • NEXT — N/A • SEQUENCE — N/A
CHECK DATAFILES	N/A — global or local
CHECKPOINT	N/A — global or local
COPY LOGFILE	N/A — N/A
DISTRIBUTED RECOVERY	N/A — enable or disable
DUMP	• DATAFILE — N/A • FLASHBACK — N/A • LOGFILE — N/A • REDO — N/A • TEMPFILE — N/A • UNDO — N/A
END SESSION	DISCONNECT SESSION — N/A KILL SESSION — N/A
FLUSH	BUFFER_CACHE — N/A GLOBAL CONTEXT — N/A REDO — *target_db_name* SHARED_POOL — N/A

Table 17-2 (Cont.) ALTER SYSTEM Command Rule Settings

clause_name	parameter_name — Parameter Value
QUIESCE	QUIESCE RESTRICTED — N/A
	UNQUIESCE — N/A
REFRESH	LDAP_REGISTRATION — N/A
REGISTER	N/A — N/A
RESET	*initialization_parameter_name* — N/A
RESUME	N/A — N/A
SECURITY	RESTRICTED SESSION — enable or disable
	SET ENCRYPTION KEY — N/A
	SET ENCRYPTION WALLET — open or close
SET	EVENTS — *event_string*
	GLOBAL_TOPIC_ENABLED — true or false
	initialization_parameter_name — *parameter_value*
	LDAP_REGISTRATION_ENABLED — true or false
	LDAP_REG-SYNC_INTERVAL — Number
	SINGLETASK DEBUG — N/A
	USE_STORED_OUTLINES — true , false, or *category_name*
SHUTDOWN DISPPATCHER	N/A — *dispatcher_name*
SWITCH LOGFILE	N/A — all or none
SUSPEND	N/A — N/A
TX RECOVERY	N/A — enable or disable

ALTER SESSION Command Rule Settings

Table 17-3 describes the ALTER SESSION command rule settings.

Table 17-3 ALTER SESSION Command Rule Settings

clause_name	parameter_name — Parameter Value
ADVISE	N/A — COMMIT, ROLLBACK, or NOTHING
CLOSE DATABASE LINK	N/A — *database_link*
COMMIT IN PROCEDURE	N/A — ENABLE or DISABLE
GUARD	N/A — ENABLE or DISABLE
ILM	ROW ACCESS TRACKING — N/A
	ROW MODIFICATION TRACKING — N/A
LOGICAL REPLICATION	N/A — N/A
PARALLEL DML	N/A — ENABLE, DISABLE, or FORCE
PARALLEL DDL	N/A — ENABLE, DISABLE, or FORCE

Table 17-3 (Cont.) ALTER SESSION Command Rule Settings

clause_name	parameter_name — Parameter Value
PARALLEL QUERY	N/A — ENABLE, DISABLE, or FORCE
RESUMABLE	N/A — ENABLE or DISABLE
SYNC WITH PRIMARY	N/A — N/A
SET	APPLICATION ACTION — *action_name* APPLICATION MODULE — *module_name* CONSTRAINTS — IMMEDIATE, DEFERRED, or DEFAULT CONTAINER — *container_name* CURRENT SCHEMA — *schema_name* EDITION — *edition_name* ERROR ON OVERLAP TIME — TRUE or FALSE EVENTS — *event_string* FLAGGER — OFF, FULL, INTERMEDIATE, ENTRY *initialization_parameter_name* — *parameter_name* INSTANCE — *instance_number* ISOLATION_LEVEL — SERIALIZABLE or READ COMMITTED ROW_ARCHIVAL_VISABILITY — ACTIVE or ALL SQL_TRANSFORMATION_PROFILE — *profile_name* STANDBY_MAX_DATA_DELAY — NONE*number* TIME_ZONE — LOCAL, DBTIMEZONE, or *other_value* USE_PRIVATE_OUTLINES — TRUE, FALSE, or *category_name* USE_STORED_OUTLINES — TRUE, FALSE, or *category_name*

Examples

The following example shows how to create a simple command rule for the SELECT statement on the OE.ORDERS table. This command rule uses no command rules.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'SELECT',
  rule_set_name   => 'Check User Role',
  object_owner    => 'OE',
  object_name     => 'ORDERS',
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

ALTER SESSION Command Rule Using the SET Clause

The following example shows how to create an ALTER SESSION command rule that uses the SET clause with the ERROR_ON_OVERLAP_TIME parameter.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'ALTER SESSION',
```

```
  rule_set_name   => 'Test ERROR_ON_OVERLAP_TIME for FALSE',
  object_owner    => '%',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES,
 clause_name     => 'SET',
 parameter_name  => 'ERROR_ON_OVERLAP_TIME',
 scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

In this example:

- `rule_set_name`: The `ALTER SESSION` SQL statement `ERROR_ON_OVERLAP_TIME` session parameter must be set to either `TRUE` or `FALSE`. You can create a rule set that checks if this setting. For example, for the rule:

  ```
  EXEC DBMS_MACADM.CREATE_RULE('RULE_TRUE', 'UPPER(PARAMETER_VALUE) = ''TRUE''');
  ```

  The rule set that is used with this rule can be similar to the following:

  ```
  BEGIN
   DBMS_MACADM.CREATE_RULE_SET(
    rule_set_name    => 'Test ERROR_ON_OVERLAP_TIME',
    description      => 'Checks if the ERROR_ON_OVERLAP_TIME setting is TRUE or
  FALSE',
    enabled          => DBMS_MACUTL.G_YES,
    eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
    audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL +
  DBMS_MACUTL.G_RULESET_AUDIT_SUCCESS,
    fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
    fail_message     => 'false error on overlaptime',
    fail_code        => 20461,
    handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
    handler          => '',
    is_static        => false);
  END;
  /
  EXEC DBMS_MACADM.ADD_RULE_TO_RULE_SET('Test ERROR_ON_OVERLAP_TIME', 'RULE_TRUE');
  ```

- `object_owner` and `object_name` must be set to `%` for `ALTER SESSION` and `ALTER SYSTEM` command rules.
- `enabled` uses the `DBMS_MACUTL.G_YES` constant to enable the command rule when it is created.
- `clause_name` sets the `ALTER SESSION` command rule to use the `SET` clause of the `ALTER SESSION` PL/SQL statement.
- `parameter_name` is set to the `ERROR_ON_OVERLAP_TIME` parameter of the `SET` clause.
- `scope` uses the `DBMS_MACUTL.G_SCOPE_COMMON` constant to set the command rule to be a common command rule. This command rule will be in the application root of a multitenant environment, so the user running this procedure must be in the CDB root. Any rules or rule sets that are associated with this command rule must be common.

  If you were creating the command rule locally, you would set `scope` to `DBMS_MACUTL.G_SCOPE_LOCAL`. In that case, the user who runs this procedure must be in the PDB in which the command rule will reside. To find the existing PDBs, you can query the `DBA_PDBS` data dictionary view. Any rules or rule sets that are associated with this command rule must be local.

ALTER SYSTEM Command Rule Using the CHECKPOINT Clause

This example shows how to create an ALTER SYSTEM command rule that users the `CHECKPOINT` clause. To have the command rule test for the `CHECKPOINT` setting, you must create a rule set and rule, similar to the ALTER SESSION command rule in the previous example. In this example, the `parameter` setting is not specified because the `CHECKPOINT` setting does not have parameters.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'ALTER SYSTEM',
  rule_set_name   => 'Test CHECKPOINT Setting',
  object_owner    => '%',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES,
 clause_name     => 'CHECKPOINT',
 parameter_name  => '',
 scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

ALTER SESSION Command Rule Using the SET Clause

The following ALTER SESSION command rule uses the `SET` clause to specify an `event_name` and `component_name`. You can only use the `event_name`, `component_name`, and `action_name` parameters if the `clause_name` parameter specifies `SET`.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'ALTER SESSION',
  rule_set_name   => 'Check Trace Events',
  object_owner    => '%',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES,
  clause_name     => 'SET',
  parameter_name  => 'EVENTS',
  event_name      => 'TRACE',
  component_name  => 'GCS',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

See also ALTER SESSION and ALTER SYSTEM Command Rules for conceptual information about this topic.

CREATE_CONNECT_COMMAND_RULE Procedure

The `CREATE_CONNECT_COMMAND_RULE` procedure creates a CONNECT command rule that you can associate with a user and a rule set.

In a multitenant environment, you can create both common and local command rules.

Syntax

```
DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
 user_name       IN VARCHAR2,
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 17-4 CREATE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
`user_name`	User to whom the CONNECT command rule will apply. If you enter the % wildcard, then the CONNECT command rule will be applied to every database user. In a multitenant environment, if you execute this procedure in the root, then specifying % applies to all common users. If you run the procedure in a PDB, then it applies to all local and common users who have access to this PDB. If there are two command rules, one common and one local, and they both apply to the same object, then both must evaluate successfully for the operation to succeed. In a multitenant environment, ensure that this user is common if the CONNECT command rule is common, and local or common if the CONNECT command rule is local. To find existing database users in the current instance, query the `DBA_USERS` view, described in *Oracle Database Reference*.
`rule_set_name`	Name of rule set to associate with this command rule. In a multitenant environment, ensure that this rule set is common if the CONNECT command rule is common, and local if the CONNECT command rule is local. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view, described in DBA_DV_RULE_SET View.
`enabled`	Specify one of the following options to set the status of the command rule: • `DBMS_MACUTL.G_YES` or `'y'` (Yes) to enable the command rule (default) • `DBMS_MACUTL.G_NO` or `'n'` to disable the command rule, including the capture of violations in the simulation log • `DBMS_MACUTL.G_SIMULATION` or `'s'` to enable SQL statements to execute but capture violations in the simulation log
`scope`	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • `DBMS_MACUTL.G_SCOPE_LOCAL` (or 1) if the command rule is local in the current PDB • `DBMS_MACUTL.G_SCOPE_COMMON` (or 2) if the command rule is in the application root If you create the common CONNECT command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`

Examples

The following example shows how to create a common CONNECT command rule in a multitenant environment. This command rule will be in the CDB root, so the user who runs this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
```

```
  rule_set_name   => 'Allow Sessions',
  user_name       => 'C##HR_ADMIN',
  enabled         => DBMS_MACUTL.G_SIMULATION,
  scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This example is a local version of the preceding example. The user who runs this procedure must be in the PDB in which the local CONNECT command rule will reside. To find the available PDBs, run the `show pdbs` command. Any rule sets that are associated with this command rule must be local. The user can be either common or local.

```
BEGIN
 DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
 rule_set_name   => 'Allow Sessions',
 user_name       => 'PSMITH',
 enabled         => DBMS_MACUTL.G_SIMULATION,
 scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

CREATE_SESSION_EVENT_CMD_RULE Procedure

The `CREATE_SESSION_EVENT_CMD_RULE` procedure creates a command rule that you can associate with session events, based on the `ALTER SESSION` statement.

In a multitenant environment, you can create both session event common and local command rules.

Syntax

```
DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE(
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT,
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 17-5 CREATE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
`rule_set_name`	Name of the rule set to associate with the command rule. In a multitenant environment, ensure that this rule set is common if the session event command rule is common, and local if the command rule is local. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view, described in DBA_DV_RULE_SET View.

Table 17-5 (Cont.) CREATE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' (Yes) to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
event_name	An event that the command rule defines. This setting enables the command rule to correspond with an ALTER SESSION SET EVENTS *event_name* statement. For example, to track trace events, you would set *event_name* to TRACE.
component_name	A component of the event_name setting. Example settings are DV, OLS, or GCS. You can find valid component names by issuing ORADEBUG DOC COMPONENT RDBMS as user SYS. The output displays parent and child components, which you can use for the component_name setting. For example, both XS (parent) and XSSESSION (child of XS) are valid component names. If you select the parent component, then the command rule applies to it and the child components.
action_name	An action of the component_name setting
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you create the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record. The default is FALSE.

Examples

The following example shows how to create a common session event command rule in a multitenant environment. This command rule will be in the application root, so the user running this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE(
  rule_set_name   => 'Allow Sessions',
  event_name      => 'TRACE',
  component_name  => 'DV',
  action_name     => 'CURSORTRACE',
```

```
  enabled         => DBMS_MACUTL.G_SIMULATION,
  scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This example shows how to create a session event for the 47998 trace event. This example will records the PL/SQL stack for failed operations.

```
BEGIN
 DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE(
  rule_set_name   => 'Allow Sessions',
  event_name      => '47998',
  enabled         => 'y',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL,
  pl_sql_stack    => TRUE);
END;
/
```

CREATE_SYSTEM_EVENT_CMD_RULE Procedure

The `CREATE_SYSTEM_EVENT_CMD_RULE` procedure creates a command rule that you can associate with system events, based on the `ALTER SYSTEM` statement.

In a multitenant environment, you can create both ALTER SYSTEM common and local command rules.

Syntax

```
DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE(
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 17-6 CREATE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
`rule_set_name`	Name of the rule set to associate with the command rule. In a multitenant environment, ensure that this rule set is common if the system event command rule is common, and local if the command rule is local. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view, described in DBA_DV_RULE_SET View.
`event_name`	An event that the command rule defines. This setting enables the command rule to correspond to an `ALTER SYSTEM SET EVENTS` *`event_name`* statement. For example, to track trace events, you would set *`event_name`* to `TRACE`.

Table 17-6 (Cont.) CREATE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
component_name	A component of the event_name setting. Example settings are DV, OLS, or GCS. You can find valid component names by issuing ORADEBUG DOC COMPONENT RDBMS as user SYS. The output displays parent and child components, which you can use for the component_name setting. For example, both XS (parent) and XSSESSION (child of XS) are valid component names. If you select the parent component, then the command rule applies to it and the child components.
action_name	An action of the component_name setting
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you create the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record. The default is FALSE.

Example

The following example shows how to create a common system event command rule in a multitenant environment. This command rule will be in the application root, so the user running this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE(
  rule_set_name   => 'Enabled',
  event_name      => 'TRACE',
  component_name  => 'GSIPC',
  action_name     => 'HEAPDUMP',
  enabled         => DBMS_MACUTL.G_YES,
  scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

DELETE_COMMAND_RULE Procedure

The DELETE_COMMAND_RULE procedure drops a command rule declaration.

Syntax

```
DBMS_MACADM.DELETE_COMMAND_RULE(
 command         IN VARCHAR2,
 object_owner    IN VARCHAR2,
 object_name     IN VARCHAR2,
 clause_name     IN VARCHAR2,
 parameter_name  IN VARCHAR2 DEFAULT,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 17-7 DELETE_COMMAND_RULE Parameters

Parameter	Description
command	SQL statement the command rule protects. To find available command rules, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View
object_owner	Database schema to which this command rule applies. To find the available users in the current database instance, query the DBA_USERS view, described in *Oracle Database Reference*.
object_name	Object name. The wildcard % is allowed. To find the available objects in the current database instance, query the ALL_OBJECTS view, described in *Oracle Database Reference*.
clause_name	A clause from the SQL statement that was used to create the command rule. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
parameter_name	A parameter from the clause_name parameter. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
event_name	An event that the command rule defines. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
component_name	A component of the event_name setting. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
action_name	An action of the component_name setting. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Example

The following example shows how to delete an ALTER SESSION command rule. When you specify the parameters, ensure that they match exactly the parameters that were used the last time the command rule was updated. To find the current settings of the command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View.

```
BEGIN
 DBMS_MACADM.DELETE_COMMAND_RULE(
  command         => 'ALTER SESSION',
  object_owner    => '%',
  object_name     => '%',
  clause_name     => 'SET',
  parameter_name  => 'EVENTS',
  event_name      => 'TRACE',
  component_name  => 'GCS',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

This example shows how to delete a SELECT command rule.

```
BEGIN
 DBMS_MACADM.DELETE_COMMAND_RULE(
  command        => 'SELECT',
  object_owner   => 'HR',
  object_name    => 'EMPLOYEES',
  scope          => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

DELETE_CONNECT_COMMAND_RULE Procedure

The DELETE_CONNECT_COMMAND_RULE procedure deletes a CONNECT command rule that had been created with the CREATE_CONNECT_COMMAND_RULE procedure.

Syntax

```
DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE(
 user_name       IN VARCHAR2,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 17-8 DELETE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
user_name	User to whom the CONNECT command rule applied. To find this user, query the OBJECT_OWNER field of the DBA_DV_COMMAND_RULE view.

Table 17-8 (Cont.) DELETE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Example

```
BEGIN
 DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE(
  user_name       => 'PSMITH',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

DELETE_SESSION_EVENT_CMD_RULE Procedure

The DELETE_SESSION_EVENT_CMD_RULE procedure deletes a session command rule that was associated with events.

Syntax

```
DBMS_MACADM.DELETE_SESSION_EVENT_CMD_RULE(
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 17-9 DELETE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
event_name	An event that the session event command rule defines. DBA_DV_COMMAND_RULE View for a information about existing command rules
component_name	A component of the event_name setting
action_name	An action of the component_name setting
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Example

The following example shows how to delete a common session event command rule in the application root a multitenant environment. The user running this procedure must be a common user in the CDB root. When you specify the parameters, ensure that they match exactly the parameters that were used the last time the command rule was updated. To find the current settings of the command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View

```
BEGIN
DBMS_MACADM.DELETE_SESSION_EVENT_CMD_RULE(
 event_name      => '47999',
 scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
 /
```

DELETE_SYSTEM_EVENT_CMD_RULE Procedure

The DELETE_SYSTEM_EVENT_CMD_RULE procedure deletes a system command rule that was associated with events.

Syntax

```
DBMS_MACADM.DELETE_SYSTEM_EVENT_CMD_RULE(
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 17-10 DELETE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
event_name	An event that the system event command rule defines. See DBA_DV_COMMAND_RULE View for a information about existing command rules.
component_name	A component of the event_name setting
action_name	An action of the component_name setting
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Examples

The following example shows how to delete a common system event command rule in the application root of a multitenant environment. The user running this procedure must be a common user in the CDB root. When you specify the parameters, ensure that they match exactly the parameters that were used the last time the command rule

was updated. To find the current settings of the command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View

```
BEGIN
 DBMS_MACADM.DELETE_SYSTEM_EVENT_CMD_RULE(
 event_name      => 'TRACE',
 component_name  => 'DV',
 action_name     => '',
 scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

UPDATE_COMMAND_RULE Procedure

The UPDATE_COMMAND_RULE procedure updates a command rule declaration.

In a multitenant environment, you can update both common and local command rules.

Syntax

```
DBMS_MACADM.UPDATE_COMMAND_RULE(
 command         IN VARCHAR2,
 rule_set_name   IN VARCHAR2,
 object_owner    IN VARCHAR2,
 object_name     IN VARCHAR2,
 enabled         IN VARCHAR2,
 privilege_scope IN NUMBER,
 clause_name     IN VARCHAR2,
 parameter_name  IN VARCHAR2 DEFAULT,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT,
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 17-11 UPDATE_COMMAND_RULE Parameters

Parameter	Description
command	Command rule to update See also the following: • SQL Statements That Can Be Protected by Command Rules for a listing of available SQL statements that you can use • DBA_DV_COMMAND_RULE View for a information about existing command rules
rule_set_name	Name of rule set to associate with this command rule. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in Oracle Database Vault Data Dictionary Views.
object_owner	Database schema to which this command rule applies. To find the available users, query the DBA_USERS view, described in *Oracle Database Reference*. See also "Object Owner" in Creating a Command Rule for more information.

Table 17-11 (Cont.) UPDATE_COMMAND_RULE Parameters

Parameter	Description
object_name	Object name. (The wildcard % is allowed. See "Object Name" in Creating a Command Rule for more information about objects protected by command rules.) To find the available objects, query the ALL_OBJECTS view, described in *Oracle Database Reference*.
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
privilege_scope	Obsolete parameter
clause_name	A clause from the SQL statement that was used to create the command rule. For example, a command rule for the ALTER SESSION SQL statement could have the SET clause as the clause_name parameter. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
parameter_name	A parameter from the clause_name parameter. For example, for an ALTER SESSION command rule, you could set parameter_name to EVENTS if the clause_name is SET. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
event_name	An event that the command rule defines. For example, for an ALTER SESSION command rule that uses SET for the clause_name and EVENTS as the parameter_name, then the event_name could be set to TRACE. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to events.
component_name	A component of the event_name setting. For example, for a TRACE event, the component_name could be GCS. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to events.
action_name	An action of the component_name setting. For example, if component_name is set to GCS, then the action_name setting could be DISK HIGH. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to events.

Table 17-11 (Cont.) UPDATE_COMMAND_RULE Parameters

Parameter	Description
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you update the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Examples

The following example shows how to create a simple command rule that protects the HR.EMPLOYEES schema.

```
BEGIN
 DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'SELECT',
  rule_set_name   => 'Enabled',
  object_owner    => 'HR',
  object_name     => 'EMPLOYEES',
  enabled         => DBMS_MACUTL.G_SIMULATION,
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

This example shows how to update a more complex command rule, which is based on the ALTER SESSION SQL statement.

```
BEGIN
 DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'ALTER SESSION',
  rule_set_name   => 'Enabled',
  object_owner    => '%',
  object_name     => '%',
  enabled         => 's',
  clause_name     => 'SET',
  parameter_name  => 'EVENTS',
  event_name      => 'TRACE',
  component_name  => 'GCS',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

UPDATE_CONNECT_COMMAND_RULE Procedure

The UPDATE_CONNECT_COMMAND_RULE procedure updates a CONNECT command rule that had been created with the CREATE_CONNECT_COMMAND_RULE procedure.

Syntax

```
DBMS_MACADM.CREATE_UPDATE_CONNECT_COMMAND_RULE(
 user_name       IN VARCHAR2,
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 17-12 UPDATE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
user_name	User to whom the CONNECT command rule will apply. If you enter the % wildcard, then the CONNECT command rule will be applied to every database user. In a multitenant environment, if you execute this procedure in the root, then specifying % applies to all common users. If you run the procedure in a PDB, then it applies to all local and common users who have access to this PDB. If there are two command rules, one common and one local, and they both apply to the same object, then both must evaluate successfully for the operation to succeed. In a multitenant environment, ensure that this user is common if the CONNECT command rule is common, and local or common if the CONNECT command rule is local. To find existing command rules, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View. To find existing database users in the current instance, query the DBA_USERS view, described in *Oracle Database Reference*.
rule_set_name	Name of rule set to associate with this command rule. In a multitenant environment, ensure that this rule set is common if the CONNECT command rule is common, and local if the CONNECT command rule is local. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log

Table 17-12 (Cont.) UPDATE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you update the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`

Example

```
BEGIN
 DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE(
  rule_set_name   => 'Allow Sessions',
  user_name       => 'PSMITH',
  enabled         => 'DBMS_MACUTL.G_YES',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

UPDATE_SESSION_EVENT_CMD_RULE Procedure

The UPDATE_SESSION_EVENT_CMD_RULE procedure updates a session event command rule, based on the ALTER SESSION statement.

In a multitenant environment, you can update both common and local session event command rules.

Syntax

```
DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE(
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT,
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 17-13 UPDATE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
rule_set_name	Name of the rule set to associate with the command rule. In a multitenant environment, ensure that this rule set is common if the session event command rule is common, and local if the command rule is local. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
event_name	An event that the command rule defines. This setting enables the command rule to correspond with an ALTER SESSION SET EVENTS *event_name* statement. For example, to track trace events, you would set *event_name* to TRACE.
component_name	A component of the event_name setting. Example settings are DV, OLS, or GCS. You can find valid component names by issuing ORADEBUG DOC COMPONENT RDBMS as user SYS. The output displays parent and child components, which you can use for the component_name setting. For example, both XS (parent) and XSSESSION (child of XS) are valid component names. If you select the parent component, then the command rule applies to it and the child components.
action_name	An action of the component_name setting
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you update the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Example

The following example shows how to update a common session event command rule in a multitenant environment. This command rule is in the application root, so the user running this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE(
  rule_set_name => 'Allow Sessions',
  event_name    => '47999',
  enabled       => DBMS_MACUTL.G_NO,
  scope         => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

UPDATE_SYSTEM_EVENT_CMD_RULE Procedure

The UPDATE_SYSTEM_EVENT_CMD_RULE procedure updates a system event command rule, based on the ALTER SYSTEM statement.

In a multitenant environment, you can update both common and local session event command rules.

Syntax

```
DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE(
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT,
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 17-14 UPDATE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
rule_set_name	Name of the rule set to associate with the command rule. In a multitenant environment, ensure that this rule set is common if the system event command rule is common, and local if the command rule is local. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
event_name	An event that the command rule defines. This setting enables the command rule to correspond to an ALTER SYSTEM SET EVENTS *event_name* statement. For example, to track trace events, you would set *event_name* to TRACE.

Table 17-14 (Cont.) UPDATE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
component_name	A component of the event_name setting. Example settings are DV, OLS, or GCS. You can find valid component names by issuing ORADEBUG DOC COMPONENT RDBMS as user SYS. The output displays parent and child components, which you can use for the component_name setting. For example, both XS (parent) and XSSESSION (child of XS) are valid component names. If you select the parent component, then the command rule applies to it and the child components.
action_name	An action of the component_name setting
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you update the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Example

The following example shows how to update a common system event command rule in a multitenant environment. This command rule is in the application root, so the user running this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE(
 rule_set_name   => 'Disabled',
 event_name      => 'TRACE',
 component_name  => 'DV',
 enabled         => 'n',
 scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

18

Oracle Database Vault Factor APIs

The DBMS_MACADM PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and DVF has functions to manage factors.

- DBMS_MACADM Factor Procedures and Functions
  The DBMS_MACADM PL/SQL package provides procedures and functions to configure factors.
- Oracle Database Vault Run-Time PL/SQL Procedures and Functions
  Oracle Database Vault provides procedural interfaces to administer Database Vault security options and manage Database Vault security enforcements.
- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the DVF schema functions when you use the DBMS_MACADM PL/SQL package to manage the various factors.

DBMS_MACADM Factor Procedures and Functions

The DBMS_MACADM PL/SQL package provides procedures and functions to configure factors.

Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures and functions.

- ADD_FACTOR_LINK Procedure
  The ADD_FACTOR_LINK procedure specifies a parent-child relationship for two factors.
- ADD_POLICY_FACTOR Procedure
  The ADD_POLICY_FACTOR procedure specifies that the label for a factor contributes to the Oracle Label Security label for a policy.
- CHANGE_IDENTITY_FACTOR Procedure
  The CHANGE_IDENTITY_FACTOR procedure associates an identity with a different factor.
- CHANGE_IDENTITY_VALUE Procedure
  The CHANGE_IDENTITY_FACTOR procedure updates the value of an identity.
- CREATE_DOMAIN_IDENTITY Procedure
  The CREATE_DOMAIN_IDENTITY procedure is used for Oracle Real Application Clusters (Oracle RAC) and Oracle Label Security.
- CREATE_FACTOR Procedure
  The CREATE_FACTOR procedure creates a factor.
- CREATE_FACTOR_TYPE Procedure
  The CREATE_FACTOR_TYPE procedure creates a user-defined factor type.
- CREATE_IDENTITY_MAP Procedure
  The CREATE_IDENTITY_MAP procedure defines tests that can derive the identity of a factor from the value of linked child factors (subfactors).

- CREATE_IDENTITY Procedure
  The CREATE_IDENTITY procedure assigns an identity and an associated trust level for a given factor.
- DELETE_FACTOR Procedure
  The DELETE_FACTOR procedure deletes a factor.
- DELETE_FACTOR_LINK Procedure
  The DELETE_FACTOR_LINK procedure removes a parent-child relationship for two factors.
- DELETE_IDENTITY Procedure
  The DELETE_IDENTITY procedure removes an identity from an existing factor.
- DELETE_FACTOR_TYPE Procedure
  The DELETE_FACTOR_TYPE procedure deletes a factor type.
- DELETE_IDENTITY_MAP Procedure
  The DELETE_IDENTITY_MAP procedure removes an identity map for a factor.
- DROP_DOMAIN_IDENTITY Procedure
  The DROP_DOMAIN_IDENTITY procedure removes an Oracle Real Application Clusters database node from a domain.
- GET_SESSION_INFO Function
  The GET_SESSION_INFO function returns information from the SYS.V_$SESSION system table for the current session.
- GET_INSTANCE_INFO Function
  The GET_INSTANCE_INFO function returns information from the SYS.V_$INSTANCE system table about the current database instance.
- RENAME_FACTOR Procedure
  The RENAME_FACTOR procedure renames a factor; the name change takes effect everywhere the factor is used.
- RENAME_FACTOR_TYPE Procedure
  The RENAME_FACTOR procedure renames a factor type; the name change takes effect everywhere the factor type is used.
- UPDATE_FACTOR Procedure
  The UPDATE_FACTOR procedure updates the description of a factor type.
- UPDATE_FACTOR_TYPE Procedure
  The UPDATE_FACTOR_TYPE procedure updates a factor type.
- UPDATE_IDENTITY Procedure
  The UPDATE_IDENTITY procedure updates the trust level of a factor identity.

Related Topics

- Configuring Factors
  Factors enable you to base Database Vault restrictions on attributes such as a client IP address or a domain.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the DBMS_MACUTL PL/SQL package.

ADD_FACTOR_LINK Procedure

The ADD_FACTOR_LINK procedure specifies a parent-child relationship for two factors.

Syntax

```
DBMS_MACADM.ADD_FACTOR_LINK(
  parent_factor_name IN VARCHAR2,
  child_factor_name  IN VARCHAR2,
  label_indicator    IN VARCHAR2);
```

Parameters

Table 18-1 ADD_FACTOR_LINK Parameters

Parameter	Description
parent_factor_name	Parent factor name. To find existing parent and child factors in the current database instance, query the DBA_DV_FACTOR_LINK view, described in DBA_DV_FACTOR_LINK View.
child_factor_name	Child factor name.
label_indicator	Indicates that the child factor being linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Specify either DBMS_MACUTL.G_YES (for Yes) or DBMS_MACUTL.G_NO (for No). To find the Oracle Label Security policies and labels associated with factors, query the following views, described in Oracle Database Vault Data Dictionary Views: • DBA_DV_MAC_POLICY: Lists Oracle Label Security policies defined in the current database instance. • DBA_DV_MAC_POLICY_FACTOR: Lists the factors that are associated with Oracle Label Security policies for the current database instance. • DBA_DV_POLICY_LABEL: Lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.

Example

```
BEGIN
 DBMS_MACADM.ADD_FACTOR_LINK(
  parent_factor_name => 'HQ_ClientID',
  child_factor_name  => 'Div1_ClientID',
  label_indicator    => DBMS_MACUTL.G_YES);
END;
/
```

ADD_POLICY_FACTOR Procedure

The ADD_POLICY_FACTOR procedure specifies that the label for a factor contributes to the Oracle Label Security label for a policy.

Syntax

```
DBMS_MACADM.ADD_POLICY_FACTOR(
  policy_name  IN VARCHAR2,
  factor_name  IN VARCHAR2);
```

Parameters

Table 18-2 ADD_POLICY_FACTOR Parameters

Parameter	Description
policy_name	Oracle Label Security policy name. To find the policies defined in the current database instance, query the DBA_DV_MAC_POLICY view, described in DBA_DV_MAC_POLICY View. To find factors that are associated with Oracle Label Security policies, query DBA_DV_MAC_POLICY_FACTOR, described in DBA_DV_MAC_POLICY View.
factor_name	Factor name. To find existing factors, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.

Example

```
BEGIN
 DBMS_MACADM.ADD_POLICY_FACTOR(
  policy_name  => 'AccessData',
  factor_name  => 'Sector2_ClientID');
END;
/
```

CHANGE_IDENTITY_FACTOR Procedure

The CHANGE_IDENTITY_FACTOR procedure associates an identity with a different factor.

Syntax

```
DBMS_MACADM.CHANGE_IDENTITY_FACTOR(
  factor_name     IN VARCHAR2,
  value           IN VARCHAR2,
  new_factor_name IN VARCHAR2);
```

Parameters

Table 18-3 CHANGE_IDENTITY_FACTOR Parameters

Parameter	Description
factor_name	Current factor name. To find existing factors, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.
value	Value of the identity to update. To find existing identities for each factor in the current database instance, query the DBA_DV_IDENTITY view, described in DBA_DV_IDENTITY View. To find current identity mappings, query the DBA_DV_IDENTITY_MAP view, described in DBA_DV_IDENTITY_MAP View.
new_factor_name	Name of the factor to associate with the identity, which you can find by querying the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.

Example

```
BEGIN
 DBMS_MACADM.CHANGE_IDENTITY_FACTOR(
  factor_name     => 'Sector2_ClientID',
  value           => 'intranet',
  new_factor_name => 'Sector4_ClientID');
END;
/
```

CHANGE_IDENTITY_VALUE Procedure

The CHANGE_IDENTITY_FACTOR procedure updates the value of an identity.

Syntax

```
DBMS_MACADM.CHANGE_IDENTITY_VALUE(
  factor_name  IN VARCHAR2,
  value        IN VARCHAR2,
  new_value    IN VARCHAR2);
```

Parameters

Table 18-4 CHANGE_IDENTITY_VALUE Parameters

Parameter	Description
factor_name	Factor name. To find existing factors, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.

Table 18-4 (Cont.) CHANGE_IDENTITY_VALUE Parameters

Parameter	Description
value	Current value associated with the identity. To find existing identities for each factor in the current database instance, query the DBA_DV_IDENTITY view, described in DBA_DV_FACTOR View. To find current identity mappings, query the DBA_DV_IDENTITY_MAP view, described in DBA_DV_IDENTITY_MAP View.
new_value	New identity value, up to 1024 characters in mixed-case.

Example

```
BEGIN
 DBMS_MACADM.CHANGE_IDENTITY_VALUE(
  factor_name  => 'Sector2_ClientID',
  value        => 'remote',
  new_value    => 'intranet');
END;
/
```

CREATE_DOMAIN_IDENTITY Procedure

The CREATE_DOMAIN_IDENTITY procedure is used for Oracle Real Application Clusters (Oracle RAC) and Oracle Label Security.

It adds an Oracle RAC database node to the domain factor identities and labels it according to an Oracle Label Security policy

Syntax

```
DBMS_MACADM.CREATE_DOMAIN_IDENTITY(
  domain_name  IN VARCHAR2,
  domain_host  IN VARCHAR2,
  policy_name  IN VARCHAR2 DEFAULT NULL,
  domain_label IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 18-5 CREATE_DOMAIN_IDENTITY Parameters

Parameter	Description
domain_name	Name of the domain to which to add the host. To find the logical location of the database within the network structure within a distributed database system, run the DVF.F$DATABASE_DOMAIN function, described in Oracle Database Vault DVF PL/SQL Factor Functions.
domain_host	Oracle Real Application Clusters host name being added to the domain. To find host name of a database, run the DVF.F$DATABASE_HOSTNAME function, described in Oracle Database Vault DVF PL/SQL Factor Functions.

Table 18-5 (Cont.) CREATE_DOMAIN_IDENTITY Parameters

Parameter	Description
policy_name	Oracle Label Security policy name. If you omit the policy name, then the domain is not associated with any policy. To find the available policies, query the DBA_DV_MAC_POLICY view, described in DBA_DV_MAC_POLICY View.
domain_label	Name of the domain to which to add the Oracle Label Security policy.

Examples

```
BEGIN
 DBMS_MACADM.CREATE_DOMAIN_IDENTITY(
  domain_name  => 'example',
  domain_host  => 'mydom_host',
  policy_name  => 'AccessData',
  domain_label => 'sensitive');
END;
/
```

CREATE_FACTOR Procedure

The CREATE_FACTOR procedure creates a factor.

After you create a factor, you can give it an identity by using the CREATE_IDENTITY procedure, described in CREATE_IDENTITY Procedure.

Syntax

```
DBMS_MACADM.CREATE_FACTOR(
  factor_name       IN VARCHAR2,
  factor_type_name  IN VARCHAR2,
  description       IN VARCHAR2,
  rule_set_name     IN VARCHAR2,
  get_expr          IN VARCHAR2,
  validate_expr     IN VARCHAR2,
  identify_by       IN NUMBER,
  labeled_by        IN NUMBER,
  eval_options      IN NUMBER,
  audit_options     IN NUMBER,
  fail_options      IN NUMBER);
```

Parameters

Table 18-6 CREATE_FACTOR Parameters

Parameter	Description
factor_name	Factor name, up to 128 characters in mixed-case, without spaces. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.

Table 18-6 (Cont.) CREATE_FACTOR Parameters

Parameter	Description
`factor_type_name`	Type of the factor, up to 128 characters in mixed-case, without spaces. To find existing factor types, query the `DBA_DV_FACTOR_TYPE` view, described in DBA_DV_FACTOR_TYPE View.
`description`	Description of the purpose of the factor, up to 1024 characters in mixed-case.
`rule_set_name`	Rule set name if you want to use a rule set to control when and how a factor identity is set. To find existing rule sets, query the `DBA_DV_RULE_SET` view, described in Oracle Database Vault Data Dictionary Views. See also Assigning a Rule Set to a Factor for more information about assigning rule sets to factors.
`get_expr`	Valid PL/SQL expression that retrieves the identity of a factor. It can use up to 255 characters in mixed-case. See Setting the Retrieval Method for a Factor for more information. See also the `audit_options` parameter.
`validate_expr`	Name of the procedure to validate the factor. This is a valid PL/SQL expression that returns a Boolean value (`TRUE` or `FALSE`) to validate the identity of the factor. See Setting the Validation Method for a Factor for more information.
`identify_by`	Options for determining the identity of a factor, based on the expression set for the `get_expr` parameter: • `DBMS_MACUTL.G_IDENTIFY_BY_CONSTANT`: By constant • `DBMS_MACUTL.G_IDENTIFY_BY_METHOD`: By method • `DBMS_MACUTL.G_IDENTIFY_BY_FACTOR`: By factor • `DBMS_MACUTL.G_IDENTIFY_BY_CONTEXT`: By context See Setting the Factor Identification Information for more information.
`labeled_by`	Options for labeling the factor: • `DBMS_MACUTL.G_LABELED_BY_SELF`: Labels the identities for the factor directly from the labels associated with an Oracle Label Security policy (default) • `DBMS_MACUTL.G_LABELED_BY_FACTORS`: Derives the factor identity label from the labels of its child factor identities. See Setting the Oracle Label Security Labeling Information for a Factor for more information.
`eval_options`	Options for evaluating the factor when the user logs on: • `DBMS_MACUTL.G_EVAL_ON_SESSION`: When the database session is created (default) • `DBMS_MACUTL.G_EVAL_ON_ACCESS`: Each time the factor is accessed • `DBMS_MACUTL.G_EVAL_ON_STARTUP`: On start-up See Setting the Evaluation Information for a Factor for more information.

Table 18-6 (Cont.) CREATE_FACTOR Parameters

Parameter	Description
audit_options	Options for auditing the factor if you want to generate a custom Oracle Database Vault audit record. • DBMS_MACUTL.G_AUDIT_OFF: Disables auditing. • DBMS_MACUTL.G_AUDIT_ALWAYS: Always audits. • DBMS_MACUTL.G_AUDIT_ON_GET_ERROR: Audits if get_expr returns an error. • DBMS_MACUTL.G_AUDIT_ON_GET_NULL: Audits if get_expr is null. • DBMS_MACUTL.G_AUDIT_ON_VALIDATE_ERROR: Audits if the validation procedure returns an error. • DBMS_MACUTL.G_AUDIT_ON_VALIDATE_FALSE: Audits if the validation procedure is false. • DBMS_MACUTL.G_AUDIT_ON_TRUST_LEVEL_NULL: Audits if there is no trust level set. • DBMS_MACUTL.G_AUDIT_ON_TRUST_LEVEL_NEG: Audits if the trust level is negative. See Setting Audit Options for a Factor for more information.
fail_options	Options for reporting factor errors: • DBMS_MACUTL.G_FAIL_WITH_MESSAGE: Shows an error message (default) • DBMS_MACUTL.G_FAIL_SILENTLY: Does not show an error message See Setting Error Options for a Factor for more information.

Example

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'Limit_DBA_Access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_METHOD,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_SESSION,
  audit_options     => DBMS_MACUTL.G_AUDIT_OFF,
  fail_options      => DBMS_MACUTL.G_FAIL_SILENTLY);
END;
/
```

CREATE_FACTOR_TYPE Procedure

The CREATE_FACTOR_TYPE procedure creates a user-defined factor type.

Syntax

```
DBMS_MACADM.CREATE_FACTOR_TYPE(
  name        IN VARCHAR2,
  description IN VARCHAR2);
```

Parameters

Table 18-7 CREATE_FACTOR_TYPE Parameters

Parameter	Description
name	Factor type name, up to 128 characters in mixed-case, without spaces. To find existing factor types, query the DBA_DV_FACTOR_TYPE view, described in DBA_DV_FACTOR_TYPE View.
description	Description of the purpose of the factor type, up to 1024 characters in mixed-case.

Example

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR_TYPE(
  name        => 'Sector2Instance',
  description => 'Checks DB instances used in Sector 2');
END;
/
```

CREATE_IDENTITY_MAP Procedure

The CREATE_IDENTITY_MAP procedure defines tests that can derive the identity of a factor from the value of linked child factors (subfactors).

Syntax

```
DBMS_MACADM.CREATE_IDENTITY_MAP(
  identity_factor_name  IN VARCHAR2,
  identity_factor_value IN VARCHAR2,
  parent_factor_name    IN VARCHAR2,
  child_factor_name     IN VARCHAR2,
  operation             IN VARCHAR2,
  operand1              IN VARCHAR2,
  operand2              IN VARCHAR2);
```

Parameters

Table 18-8 CREATE_IDENTITY_MAP Parameters

Parameter	Description
identity_factor_name	Factor the identity map is for. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.
identity_factor_value	Value the factor assumes if the identity map evaluates to TRUE. To find existing factor identities, query the DBA_DV_IDENTITY view, described in DBA_DV_IDENTITY View. To find current factor identity mappings, use DBA_DV_IDENTITY_MAP, described in DBA_DV_IDENTITY_MAP View.

Table 18-8 (Cont.) CREATE_IDENTITY_MAP Parameters

Parameter	Description
parent_factor_name	The parent factor link to which the map is related. To find existing parent-child factor mappings, query the DBA_DV_IDENTITY_MAP view, described in DBA_DV_IDENTITY_MAP View.
child_factor_name	The child factor link to which the map is related.
operation	Relational operator for the identity map (for example, <, >, =, and so on).
operand1	Left operand for the relational operator; refers to the low value you enter.
operand2	Right operand for the relational operator; refers to the high value you enter.

Example

```
BEGIN
 DBMS_MACADM.CREATE_IDENTITY_MAP(
  identity_factor_name  => 'Sector2_ClientID',
  identity_factor_value => 'intranet',
  parent_factor_name    => 'HQ_ClientID',
  child_factor_name     => 'Div1_ClientID',
  operation             => '<',
  operand1              => '192.0.2.50',
  operand2              => '192.0.2.100');
END;
/
```

CREATE_IDENTITY Procedure

The CREATE_IDENTITY procedure assigns an identity and an associated trust level for a given factor.

After you create a factor, you must assign it an identity.

Syntax

```
DBMS_MACADM.CREATE_IDENTITY(
  factor_name  IN VARCHAR2,
  value        IN VARCHAR2,
  trust_level  IN NUMBER);
```

Parameters

Table 18-9 CREATE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.

Table 18-9 (Cont.) CREATE_IDENTITY Parameters

Parameter	Description
value	The actual value of the factor, up to 1024 characters in mixed-case. For example, the identity of an IP_Address factor could be the IP address of 192.0.2.12.
trust_level	Number that indicates the magnitude of trust relative to other identities for the same factor. In general, the higher the trust level number is set, the greater the trust. A trust level of 10 indicates "very trusted." Negative trust levels are not trusted. See Creating and Configuring a Factor Identity for more information about trust levels and label security.

Example

```
BEGIN
 DBMS_MACADM.CREATE_IDENTITY(
  factor_name  => 'Sector2_ClientID',
  value        => 'intranet',
  trust_level  => 5);
END;
/
```

DELETE_FACTOR Procedure

The DELETE_FACTOR procedure deletes a factor.

Syntax

```
DBMS_MACADM.DELETE_FACTOR(
  factor_name IN VARCHAR2);
```

Parameters

Table 18-10 DELETE_FACTOR Parameter

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.

Example

```
EXEC DBMS_MACADM.DELETE_FACTOR('Sector2_ClientID');
```

DELETE_FACTOR_LINK Procedure

The DELETE_FACTOR_LINK procedure removes a parent-child relationship for two factors.

Syntax

```
DBMS_MACADM.DELETE_FACTOR_LINK(
  parent_factor_name IN VARCHAR2,
  child_factor_name  IN VARCHAR2);
```

Parameters

Table 18-11 DELETE_FACTOR_LINK Parameters

Parameter	Description
parent_factor_name	Factor name. To find factors that are used in parent-child mappings in the current database instance, query the DBA_DV_FACTOR_LINK view, described in DBA_DV_FACTOR_LINK View.
child_factor_name	Factor name

Example

```
BEGIN
 DBMS_MACADM.DELETE_FACTOR_LINK(
  parent_factor_name => 'HQ_ClientID',
  child_factor_name  => 'Div1_ClientID');
END;
/
```

DELETE_IDENTITY Procedure

The DELETE_IDENTITY procedure removes an identity from an existing factor.

Syntax

```
DBMS_MACADM.DELETE_IDENTITY(
  factor_name IN VARCHAR2,
  value       IN VARCHAR2);
```

Parameters

Table 18-12 DELETE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.

Table 18-12 (Cont.) DELETE_IDENTITY Parameters

Parameter	Description
value	Identity value associated with the factor. To find the identities for each factor in the current database instance, query the DBA_DV_IDENTITY view, described in DBA_DV_IDENTITY View.

Example

```
BEGIN
 DBMS_MACADM.DELETE_IDENTITY(
  factor_name => 'Sector2_ClientID',
  value       => 'intranet');
END;
/
```

DELETE_FACTOR_TYPE Procedure

The DELETE_FACTOR_TYPE procedure deletes a factor type.

Syntax

```
DBMS_MACADM.DELETE_FACTOR_TYPE(
  name IN VARCHAR2);
```

Parameters

Table 18-13 DELETE_FACTOR_TYPE Parameters

Parameter	Description
name	Factor type name. To find existing factor types, query the DBA_DV_FACTOR_TYPE view, described in DBA_DV_FACTOR_TYPE View.

Example

```
EXEC DBMS_MACADM.DELETE_FACTOR_TYPE('Sector2Instance');
```

DELETE_IDENTITY_MAP Procedure

The DELETE_IDENTITY_MAP procedure removes an identity map for a factor.

Syntax

```
DBMS_MACADM.DELETE_IDENTITY_MAP(
  identity_factor_name  IN VARCHAR2,
  identity_factor_value IN VARCHAR2,
  parent_factor_name    IN VARCHAR2,
  child_factor_name     IN VARCHAR2,
  operation             IN VARCHAR2,
  operand1              IN VARCHAR2,
  operand2              IN VARCHAR2);
```

Parameters

Table 18-14 DELETE_IDENTITY_MAP Parameters

Parameter	Description
identity_factor_name	Factor the identity map is for. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.
identity_factor_value	Value the factor assumes if the identity map evaluates to TRUE. To find existing factor identities, query the DBA_DV_IDENTITY view, described in DBA_DV_IDENTITY View. To find current factor identity mappings, query DBA_DV_IDENTITY_MAP, described in DBA_DV_IDENTITY_MAP View.
parent_factor_name	The parent factor link to which the map is related. To find existing parent-child factors, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR_LINK View.
child_factor_name	The child factor to which the map is related.
operation	Relational operator for the identity map (for example, <, >, =, and so on).
operand1	Left (low value) operand for the relational operator.
operand2	Right (high value) operand for the relational operator.

Example

```
BEGIN
 DBMS_MACADM.DELETE_IDENTITY_MAP(
  identity_factor_name  => 'Sector2_ClientID',
  identity_factor_value => 'intranet',
  parent_factor_name    => 'HQ_ClientID',
  child_factor_name     => 'Div1_ClientID',
  operation             => '<',
  operand1              => '192.0.2.10',
  operand2              => '192.0.2.15');
END;
/
```

DROP_DOMAIN_IDENTITY Procedure

The DROP_DOMAIN_IDENTITY procedure removes an Oracle Real Application Clusters database node from a domain.

Syntax

```
DBMS_MACADM.DROP_DOMAIN_IDENTITY(
  domain_name  IN VARCHAR2,
  domain_host  IN VARCHAR2);
```

Parameters

Table 18-15 DROP_DOMAIN_IDENTITY Parameters

Parameter	Description
domain_name	Name of the domain to which the host was added. To find the domain of a database as specified by the DB_DOMAIN initialization parameter, run the DVF.F$DATABASE_DOMAIN function, described in F$DATABASE_DOMAIN Function.
domain_host	Oracle Real Application Clusters host name being that was added to the domain. To find the host name for a specified database, run the DVF.F$DATABASE_HOSTNAME function, described in F$DATABASE_NAME Function.

Example

```
BEGIN
 DBMS_MACADM.DROP_DOMAIN_IDENTITY(
  domain_name  => 'example',
  domain_host  => 'mydom_host');
END;
/
```

GET_SESSION_INFO Function

The GET_SESSION_INFO function returns information from the SYS.V_$SESSION system table for the current session.

The V$SESSION data dictionary view also contains session information from this table. See *Oracle Database Reference* for more information.

Syntax

```
DBMS_MACADM.GET_SESSION_INFO(
  p_parameter IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 18-16 GET_SESSION_INFO Parameter

Parameter	Description
p_parameter	Column name in the SYS.V_$SESSION system table.

Example

```
DECLARE
 session_var varchar2 := null;
BEGIN
 session_var = DBMS_MACADM.GET_SESSION_INFO('PROCESS');
END;
/
```

GET_INSTANCE_INFO Function

The GET_INSTANCE_INFO function returns information from the SYS.V_$INSTANCE system table about the current database instance.

The V$INSTANCE data dictionary view also contains database instance information from this table. See *Oracle Database Reference* for more information.

Syntax

```
DBMS_MACADM.GET_INSTANCE_INFO(
  p_parameter IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 18-17 GET_INSTANCE_INFO Parameter

Parameter	Description
p_parameter	Column name in the SYS.V_$INSTANCE system table

Example

```
DECLARE
 instance_var varchar2 := null;
BEGIN
 instance_var = DBMS_MACADM.GET_INSTANCE_INFO('INSTANCE_NAME');
END;
/
```

RENAME_FACTOR Procedure

The RENAME_FACTOR procedure renames a factor; the name change takes effect everywhere the factor is used.

Syntax

```
DBMS_MACADM.RENAME_FACTOR(
  factor_name     IN VARCHAR2,
  new_factor_name IN VARCHAR2);
```

Parameters

Table 18-18 RENAME_FACTOR Parameters

Parameter	Description
factor_name	Current factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.
new_factor_name	New factor name, up to 128 characters in mixed-case, without spaces.

Example

```
BEGIN
 DBMS_MACADM.RENAME_FACTOR(
  factor_name     => 'Sector2_ClientID',
  new_factor_name => 'Sector2_Clients');
END;
/
```

RENAME_FACTOR_TYPE Procedure

The RENAME_FACTOR procedure renames a factor type; the name change takes effect everywhere the factor type is used.

Syntax

```
DBMS_MACADM.RENAME_FACTOR_TYPE(
  old_name  IN VARCHAR2,
  new_name  IN VARCHAR2);
```

Parameters

Table 18-19 RENAME_FACTOR_TYPE Parameters

Parameter	Description
old_name	Current factor type name. To find existing factor types in the current database instance, query the DBA_DV_FACTOR_TYPE view, described in DBA_DV_FACTOR_TYPE View.
new_name	New factor type name, up to 128 characters in mixed-case, without spaces.

Example

```
BEGIN
 DBMS_MACADM.RENAME_FACTOR_TYPE(
  old_name  => 'Sector2Instance',
  new_name  => 'Sector2DBInstance');
END;
/
```

UPDATE_FACTOR Procedure

The UPDATE_FACTOR procedure updates the description of a factor type.

Syntax

```
DBMS_MACADM.UPDATE_FACTOR(
  factor_name       IN VARCHAR2,
  factor_type_name  IN VARCHAR2,
  description       IN VARCHAR2,
  rule_set_name     IN VARCHAR2,
  get_expr          IN VARCHAR2,
  validate_expr     IN VARCHAR2,
  identify_by       IN NUMBER,
  labeled_by        IN NUMBER,
```

```
eval_options      IN NUMBER,
audit_options     IN NUMBER,
fail_options      IN NUMBER);
```

Parameters

Table 18-20 UPDATE_FACTOR

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.
factor_type_name	Factor type name. To find existing factor types, query the DBA_DV_FACTOR_TYPE view, described in DBA_DV_FACTOR_TYPE View.
description	Description of the purpose of the factor, up to 1024 characters in mixed-case.
rule_set_name	Name of the rule set used to control when and how a factor identity is set. To find existing rule sets, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View. See also Assigning a Rule Set to a Factor for more information about assigning rule sets to factors.
get_expr	Valid PL/SQL expression that retrieves the identity of a factor. It can use up to 255 characters in mixed-case. See Setting the Retrieval Method for a Factor for more information. See also the audit_options parameter.
validate_expr	Name of the procedure to validate factor. This is a valid PL/SQL expression that returns a Boolean value (TRUE or FALSE) to validate the identity of the factor. See Setting the Validation Method for a Factor for more information.
identify_by	Options for determining the identity of a factor, based on the expression set for the get_expr parameter: • DBMS_MACUTL.G_IDENTIFY_BY_CONSTANT: By constant • DBMS_MACUTL.G_IDENTIFY_BY_METHOD: By method • DBMS_MACUTL.G_IDENTIFY_BY_FACTOR: By factor • DBMS_MACUTL.G_IDENTIFY_BY_CONTEXT: By context See Setting the Factor Identification Information for more information.
labeled_by	Options for labeling the factor: • DBMS_MACUTL.G_LABELED_BY_SELF: Labels the identities for the factor directly from the labels associated with an Oracle Label Security policy • DBMS_MACUTL.G_LABELED_BY_FACTORS: Derives the factor identity label from the labels of its child factor identities. The default for labeled_by is the previously set value, which you can find by querying the DBA_DV_FACTOR data dictionary view. See Setting the Oracle Label Security Labeling Information for a Factor for more information.

Table 18-20 (Cont.) UPDATE_FACTOR

Parameter	Description
eval_options	Options for evaluating the factor when the user logs on: • DBMS_MACUTL.G_EVAL_ON_SESSION: When the database session is created • DBMS_MACUTL.G_EVAL_ON_ACCESS: Each time the factor is accessed • DBMS_MACUTL.G_EVAL_ON_STARTUP: On start-up The default for eval_options is the previously set value, which you can find by querying the DBA_DV_FACTOR data dictionary view. See Setting the Evaluation Information for a Factor for more information.
audit_options	Options for auditing the factor if you want to generate a custom Oracle Database Vault audit record. • DBMS_MACUTL.G_AUDIT_OFF: Disables auditing. • DBMS_MACUTL.G_AUDIT_ALWAYS: Always audits. • DBMS_MACUTL.G_AUDIT_ON_GET_ERROR: Audits if get_expr returns an error. • DBMS_MACUTL.G_AUDIT_ON_GET_NULL: Audits if get_expr is null. • DBMS_MACUTL.G_AUDIT_ON_VALIDATE_ERROR: Audits if the validation procedure returns an error. • DBMS_MACUTL.G_AUDIT_ON_VALIDATE_FALSE: Audits if the validation procedure is false. • DBMS_MACUTL.G_AUDIT_ON_TRUST_LEVEL_NULL: Audits if there is no trust level set. • DBMS_MACUTL.G_AUDIT_ON_TRUST_LEVEL_NEG: Audits if the trust level is negative. The default for audit_options is the previously set value, which you can find by querying the DBA_DV_FACTOR data dictionary view. See Setting Audit Options for a Factor for more information.
fail_options	Options for reporting factor errors: • DBMS_MACUTL.G_FAIL_WITH_MESSAGE: Shows an error message. • DBMS_MACUTL.G_FAIL_SILENTLY: Does not show an error message. The default for fail_options is the previously set value, which you can find by querying the DBA_DV_FACTOR data dictionary view. See Setting Error Options for a Factor for more information.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'Limit_DBA_Access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_METHOD,
```

```
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_ACCESS,
  audit_options     => DBMS_MACUTL.G_AUDIT_ALWAYS,
  fail_options      => DBMS_MACUTL.G_FAIL_WITH_MESSAGE);
END;
/
```

UPDATE_FACTOR_TYPE Procedure

The UPDATE_FACTOR_TYPE procedure updates a factor type.

Syntax

```
DBMS_MACADM.UPDATE_FACTOR_TYPE(
  name         IN VARCHAR2,
  description  IN VARCHAR2);
```

Parameters

Table 18-21 UPDATE_FACTOR_TYPE Parameters

Parameter	Description
name	Factor type name. To find existing factor types in the current database instance, query the DBA_DV_FACTOR_TYPE view, described in DBA_DV_FACTOR_TYPE View.
description	Description of the purpose of the factor type, up to 1024 characters in mixed case.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_FACTOR_TYPE(
  name        => 'Sector2DBInstance',
  description => 'Checks DB instances used in Sector 2');
END;
/
```

UPDATE_IDENTITY Procedure

The UPDATE_IDENTITY procedure updates the trust level of a factor identity.

Syntax

```
DBMS_MACADM.UPDATE_IDENTITY(
  factor_name  IN VARCHAR2,
  value        IN VARCHAR2,
  trust_level  IN NUMBER);
```

Parameters

Table 18-22 UPDATE_IDENTITY Parameters

Parameter	Description
`factor_name`	Factor name. To find existing factors in the current database instance, query the `DBA_DV_FACTOR` view, described in DBA_DV_FACTOR View. To find factors that have identities, query `DBA_DV_IDENTITY`, described in DBA_DV_IDENTITY View.
`value`	New factor identity, up to 1024 characters in mixed-case. For example, the identity of an IP_Address factor could be the IP address of 192.0.2.12.
`trust_level`	Number that indicates the magnitude of trust relative to other identities for the same factor. In general, the higher the trust level number is set, the greater the trust. A trust level of 10 indicates "very trusted." Negative trust levels are not trusted. See Creating and Configuring a Factor Identity for more information about trust levels and label security.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_IDENTITY(
  factor_name  => 'Sector2_ClientID',
  value        => 'intranet',
  trust_level  => 10);
END;
/
```

Oracle Database Vault Run-Time PL/SQL Procedures and Functions

Oracle Database Vault provides procedural interfaces to administer Database Vault security options and manage Database Vault security enforcements.

- About Oracle Database Vault Run-Tine PL/SQL Procedures and Functions
  Oracle Database Vault provides a set of PL/SQL procedures and functions that are specific to factors.
- SET_FACTOR Procedure
  The `SET_FACTOR` procedure can be exposed to an application that requires the ability to set factor identities dynamically.
- GET_FACTOR Function
  The `GET_FACTOR` function is exposed to the `DVF` schema to allow the public factor functions to resolve the identity of a factor. The return type is `VARCHAR2`.
- GET_FACTOR_LABEL Function
  The `GET_FACTOR_LABEL` function returns the label for the specified factor when the factor has a label assigned to it for the specified Oracle Label Security policy. The return type is `VARCHAR2`.

- GET_TRUST_LEVEL Function
  The GET_TRUST_LEVEL function returns the trust level of the current session identity for the factor requested. The return type is VARCHAR2.
- GET_TRUST_LEVEL_FOR_IDENTITY Function
  The GET_TRUST_LEVEL_FOR_IDENTITY function returns the trust level for the factor and identity requested. The return type is VARCHAR2.
- ROLE_IS_ENABLED Function
  The ROLE_IS_ENABLED function returns a boolean value that specifies whether a database role has been enabled. The return type is BOOLEAN.

About Oracle Database Vault Run-Tine PL/SQL Procedures and Functions

Oracle Database Vault provides a set of PL/SQL procedures and functions that are specific to factors.

These procedures and functions that expose the logic to validate a DDL command for realm violations and command authorizations. Additional procedures and functions are provided to set the value of a factor (assuming their associated rule sets evaluate to true) (for example, from a Web application), to retrieve the trust level for a session or specific factor identity, and to get the label for a factor identity. These procedures and functions are provided so that a database administrator does not grant the EXECUTE privilege on all DVSYS package procedures to the general database account population. The procedures and functions expose only the minimum methods that are required. All of these functions and procedures are publicly available for applications that need them.

SET_FACTOR Procedure

The SET_FACTOR procedure can be exposed to an application that requires the ability to set factor identities dynamically.

It wraps the package procedure DBMS_MACADM.SET_FACTOR. When a factor has a rule set associated with it for assignment and if the rule set returns true, then the value is set. Normal rule set handling occurs, and the factor value (identity) validation method is called. This procedure is available (to execute) to the general database account population.

Syntax

```
SET_FACTOR(
  p_factor IN VARCHAR2,
  p_value  IN VARCHAR2);
```

Parameters

Table 18-23 SET_FACTOR Parameters

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR data dictionary view, described in DBA_DV_FACTOR View.
p_value	Identity value, up to 1024 characters in mixed case. To find the identities for each factor in the current database instance, query the DBA_DV_IDENTITY data dictionary view, described in DBA_DV_IDENTITY View.

Example

```
EXECUTE SET_FACTOR(''Sector2_ClientID'', ''identity'');
```

GET_FACTOR Function

The GET_FACTOR function is exposed to the DVF schema to allow the public factor functions to resolve the identity of a factor. The return type is VARCHAR2.

This function enables the F$ functions in the DVF schema. This function is available (to execute) to the general database account population.

Syntax

```
GET_FACTOR(
  p_factor IN VARCHAR2)
RETURN VARCHAR2;
```

Parameter

Table 18-24 GET_FACTOR Parameter

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR data dictionary view, described in DBA_DV_FACTOR View.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Factor Identity',
  rule_expr => 'GET_FACTOR(''Sector2_ClientID'')');
END;
/
```

GET_FACTOR_LABEL Function

The GET_FACTOR_LABEL function returns the label for the specified factor when the factor has a label assigned to it for the specified Oracle Label Security policy. The return type is VARCHAR2.

The function returns a label that is merged with the maximum session label for the policy if the policy is configured with Oracle Label Security. The function is available (to execute) to the general database population.

Syntax

```
GET_FACTOR_LABEL(
  p_factor      IN VARCHAR2,
  p_policy_name IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 18-25 GET_FACTOR_LABEL Parameters

Parameter	Description
p_factor	Factor name. To find the available factors in the current database instance, query the DBA_DV_FACTOR data dictionary view. To find factors that are associated with Oracle Label Security policies, use DBA_DV_MAC_POLICY_FACTOR. See DBA_DV_FACTOR View and DBA_DV_MAC_POLICY_FACTOR View.
p_policy_name	Oracle Label Security policy name. Use the following data dictionary views to find information about policies and factors in the current database instance: • DBA_DV_MAC_POLICY: Lists Oracle Label Security policies defined in the current database instance. See DBA_DV_MAC_POLICY View. • DBA_DV_MAC_POLICY_FACTOR: Lists the factors that are associated with Oracle Label Security policies for the current database instance. See DBA_DV_MAC_POLICY_FACTOR View. • DBA_DV_POLICY_LABEL: Lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy. See DBA_DV_POLICY_LABEL View.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get the ClientID Factor Label',
  rule_expr => 'GET_FACTOR_LABEL(''Sector2_ClientID'', ''Access Locations'')');
END;
/
```

GET_TRUST_LEVEL Function

The GET_TRUST_LEVEL function returns the trust level of the current session identity for the factor requested. The return type is VARCHAR2.

This function is available (to execute) to the general database account population. See Creating and Configuring a Factor Identity for a listing of the available trust levels.

Syntax

```
GET_TRUST_LEVEL(
  p_factor IN VARCHAR2)
RETURN VARCHAR2;
```

Parameter

Table 18-26 GET_TRUST_LEVEL Parameter

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR data dictionary view, described in DBA_DV_FACTOR View.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Trust Level',
  rule_expr => 'GET_TRUST_LEVEL(''Sector2_ClientID'')');
END;
/
```

GET_TRUST_LEVEL_FOR_IDENTITY Function

The GET_TRUST_LEVEL_FOR_IDENTITY function returns the trust level for the factor and identity requested. The return type is VARCHAR2.

This function is available (to execute) to the general database account population. See Creating and Configuring a Factor Identity for a listing of the available trust levels.

Syntax

```
GET_TRUST_LEVEL_FOR_IDENTITY(
  p_factor   IN VARCHAR2,
  p_identity IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 18-27 GET_TRUST_LEVEL_FOR_IDENTITY Parameters

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View.
p_identity	Identity value. To find the identities for each factor in the current database instance, use the DBA_DV_IDENTITY data dictionary view, described in DBA_DV_IDENTITY View.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Identity Trust Level',
  rule_expr => 'GET_TRUST_LEVEL_FOR_IDENTITY(''Sector2_ClientID'', ''identity'')');
END;
/
```

ROLE_IS_ENABLED Function

The ROLE_IS_ENABLED function returns a boolean value that specifies whether a database role has been enabled. The return type is BOOLEAN.

This function is available (to execute) to the general database account population.

Syntax

```
ROLE_IS_ENABLED(
  p_role IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 18-28 ROLE_IS_ENABLED Parameter

Parameter	Description
p_role	Database role name to check. To find existing roles, use the following data dictionary views: • DBA_ROLES: Finds available roles in the current database instance. See *Oracle Database Reference*. • DBA_DV_REALM_AUTH: Finds the authorization of a particular role. See DBA_DV_REALM View. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See DBA_DV_ROLE View.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
```

```
  rule_name => 'Check if SYSADM Role Is Enabled',
  rule_expr => 'ROLE_IS_ENABLED(''SYSADM'')');
END;
/
```

Oracle Database Vault DVF PL/SQL Factor Functions

Oracle Database Vault maintains the DVF schema functions when you use the DBMS_MACADM PL/SQL package to manage the various factors.

- About Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault provides DVF factor-specific functions for frequently used activities.
- F$AUTHENTICATION_METHOD Function
  The F$AUTHENTICATION_METHOD function returns the method of authentication in VARCHAR2 data type.
- F$CLIENT_IP Function
  The F$CLIENT_IP function returns the IP address of the computer from which the client is connected, in VARCHAR2 data type.
- F$DATABASE_DOMAIN Function
  The F$DATABASE_DOMAIN function returns the domain of the database as specified in the DB_DOMAIN initialization parameter, in VARCHAR2 data type.
- F$DATABASE_HOSTNAME Function
  The F$DATABASE_HOSTNAME function returns the host name of the computer on which the instance is running, in VARCHAR2 data type.
- F$DATABASE_INSTANCE Function
  The F$DATABASE_INSTANCE function returns the instance identification number of the current database instance, in VARCHAR2 data type.
- F$DATABASE_IP Function
  The F$DATABASE_IP function returns the IP address of the computer on which the database instance is running, in VARCHAR2 data type.
- F$DATABASE_NAME Function
  The F$DATABASE_NAME function returns the name of the database as specified in the DB_NAME initialization parameter, in VARCHAR2 data type.
- F$DOMAIN Function
  The F$DOMAIN function returns a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level. The return type is VARCHAR2.
- FDVCLIENT_IDENTIFIER Function
  The FDVCLIENT_IDENTIFIER function returns an Oracle Database Vault client identifier.
- FDVDBLINK_INFO Function
  The FDVDBLINK_INFO function returns information about an Oracle Database Vault database link.
- FDVMODULE Function
  The FDVMODULE function returns information about an Oracle Database Vault module.

- F$ENTERPRISE_IDENTITY Function
  The F$ENTERPRISE_IDENTITY function returns the enterprise-wide identity for a user, in VARCHAR2 data type.
- F$IDENTIFICATION_TYPE Function
  The F$IDENTIFICATION_TYPE function returns the way the schema of a user was created in the database. Specifically, it reflects the IDENTIFIED clause in the CREATE/ALTER USER syntax. The return type is VARCHAR2.
- F$LANG Function
  The F$LANG function returns the ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter, for the session of the user. The return type is VARCHAR2.
- F$LANGUAGE Function
  The F$LANGUAGE function returns the language and territory currently used by a user session, along with the database character set. The return type is VARCHAR2.
- F$MACHINE Function
  The F$MACHINE function returns the computer (host) name for the database client that established the database session. The return type is VARCHAR2.
- F$NETWORK_PROTOCOL Function
  The F$NETWORK_PROTOCOL function returns the network protocol being used for communication, as specified in the PROTOCOL=*protocol* portion of the connect string. The return type is VARCHAR2.
- F$PROXY_ENTERPRISE_IDENTITY Function
  The F$PROXY_ENTERPRISE_IDENTITY function returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user. The return type is VARCHAR2.
- F$PROXY_USER Function
  The F$PROXY_USER function returns the name of a proxy user.
- F$SESSION_USER Function
  The F$SESSION_USER function returns the database user name by which the current user is authenticated. This value remains the same throughout the session. The return type is VARCHAR2.

About Oracle Database Vault DVF PL/SQL Factor Functions

Oracle Database Vault provides DVF factor-specific functions for frequently used activities.

In addition to the functions and procedures made available from the DVSYS schema, the DVF schema contains a single function for each factor defined in the system.

The functions are then available to the general database account population through PL/SQL functions and standard SQL. This enables factors to be used in Oracle Label Security, Oracle Virtual Private Database (VPD), and so on.

Typically, you can incorporate these functions into rule expressions. For example:

The functions are then available to the general database account population through PL/SQL functions and standard SQL. This enables factors to be used in Oracle Label Security, Oracle Virtual Private Database (VPD), and so on.

Typically, you can incorporate these functions into rule expressions. For example:

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
 rule_name => 'Not Internal DBA',
 rule_expr => 'DVF.F$SESSION_USER NOT IN (''JSMTIH'', ''TBROWN'')');
END;
/
```

To find the value of a factor function, select from the DUAL system table. For example:

```
SELECT DVF.F$SESSION_USER FROM DUAL;

F$SESSION_USER
-----------------------------------------------
LEO_DVOWNER
```

The name of the factor itself is case-insensitive. For example, the following statements return the same result

```
select dvf.f$session_user from dual;

SELECT DVF.F$SESSION_USER FROM DUAL;
```

F$AUTHENTICATION_METHOD Function

The F$AUTHENTICATION_METHOD function returns the method of authentication in VARCHAR2 data type.

In the list that follows, the type of user is followed by the method returned:

- Password-authenticated enterprise user, local database user, or SYSDBA/SYSOPER using Password File; proxy with user name using password: PASSWORD
- Kerberos-authenticated enterprise or external user: KERBEROS
- SSL-authenticated enterprise or external user: SSL
- Radius-authenticated external user: RADIUS
- Operating system-authenticated external user or SYSDBA/SYSOPER: OS
- DCE-authenticated external user: DCE
- Proxy with certificate, distinguished name (DN), or user name without using password: NONE

You can use IDENTIFICATION_TYPE to distinguish between external and enterprise users when the authentication method is Password, Kerberos, or SSL.

Syntax

```
DVF.F$AUTHENTICATION_METHOD ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check SSL Authentication Method',
```

```
  rule_expr => 'DVF.F$AUTHENTICATION_METHOD = ''SSL''');
END;
/
```

F$CLIENT_IP Function

The `F$CLIENT_IP` function returns the IP address of the computer from which the client is connected, in `VARCHAR2` data type.

Syntax

```
DVF.F$CLIENT_IP ()
RETURN VARCHAR2;
```

Parameters

None

Example

The following example shows how to use `DVF.F$CLIENT_IP` in a rule creation statement. Note that you can only enter one IP address, not a range of IP addresses.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client IP Address',
  rule_expr => 'DVF.F$CLIENT_IP = ''192.0.2.10''');
END;
/
```

F$DATABASE_DOMAIN Function

The `F$DATABASE_DOMAIN` function returns the domain of the database as specified in the `DB_DOMAIN` initialization parameter, in `VARCHAR2` data type.

Syntax

```
DVF.F$DATABASE_DOMAIN ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client Database Domain',
  rule_expr => 'DVF.F$DATABASE_DOMAIN NOT IN (''EXAMPLE'', ''YOURDOMAIN'')');
END;
/
```

F$DATABASE_HOSTNAME Function

The F$DATABASE_HOSTNAME function returns the host name of the computer on which the instance is running, in VARCHAR2 data type.

Syntax

```
DVF.F$DATABASE_HOSTNAME ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Host Name',
  rule_expr => 'DVF.F$DATABASE_HOSTNAME IN (''SHOBEEN'', ''MAU'')');
END;
/
```

F$DATABASE_INSTANCE Function

The F$DATABASE_INSTANCE function returns the instance identification number of the current database instance, in VARCHAR2 data type.

Syntax

```
DVF.F$DATABASE_INSTANCE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database Instance ID',
  rule_expr => 'DVF.F$DATABASE_INSTANCE = ''SALES_DB''');
END;
/
```

F$DATABASE_IP Function

The F$DATABASE_IP function returns the IP address of the computer on which the database instance is running, in VARCHAR2 data type.

Syntax

```
DVF.F$DATABASE_IP ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database IP address',
  rule_expr => 'DVF.F$DATABASE_IP = ''192.0.2.5''');
END;
/
```

F$DATABASE_NAME Function

The `F$DATABASE_NAME` function returns the name of the database as specified in the `DB_NAME` initialization parameter, in `VARCHAR2` data type.

Syntax

```
DVF.F$DATABASE_NAME ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database DB_NAME Name',
  rule_expr => 'DVF.F$DATABASE_NAME = ''ORCL''');
END;
/
```

F$DOMAIN Function

The `F$DOMAIN` function returns a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level. The return type is `VARCHAR2`.

You can identify a domain using factors such as host name, IP address, and database instance names of the Oracle Database Vault nodes in a secure access path to the database. Each domain can be uniquely determined using a combination of the factor identifiers that identify the domain. You can use these identifying factors and possibly additional factors to define the Maximum Security Label within the domain. This restricts data access and commands, depending on the physical factors about the Oracle Database Vault session. Example domains of interest may be Corporate Sensitive, Internal Public, Partners, and Customers.

Syntax

```
DVF.F$DOMAIN ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Domain',
  rule_expr => 'DVF.F$DOMAIN = ''EXAMPLE.COM''');
END;
/
```

FDVCLIENT_IDENTIFIER Function

The `F$DV$CLIENT_IDENTIFIER` function returns an Oracle Database Vault client identifier.

Syntax

```
DVF.F$DV$CLIENT_IDENTIFIER ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check Database Vault Client Identifiers',
   rule_expr => 'DVF.F$DV$CLIENT_IDENTIFIER = ''14903BUA765454''';
END;/
```

FDVDBLINK_INFO Function

The `F$DV$DBLINK_INFO` function returns information about an Oracle Database Vault database link.

Syntax

```
DVF.F$DV$DBLINK_INFO ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check Database Vault database link info',
   rule_expr => 'DVF.F$DV$DBLINK_INFO = ''TBA''';
END;/
```

FDVMODULE Function

The FDVMODULE function returns information about an Oracle Database Vault module.

Syntax

```
DVF.F$DV$MODULE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check Database Vault modules',
   rule_expr => 'DVF.F$DV$MODULE = ''sqlplus''';
END;/
```

F$ENTERPRISE_IDENTITY Function

The F$ENTERPRISE_IDENTITY function returns the enterprise-wide identity for a user, in VARCHAR2 data type.

- For enterprise users: the Oracle Internet Directory DN.
- For external users: the external identity (Kerberos principal name, Radius and DCE schema names, operating system user name, certificate DN).
- For local users and SYSDBA/SYSOPER logins: NULL.

The value of the attribute differs by proxy method:

- For a proxy with DN: the Oracle Internet Directory DN of the client.
- For a proxy with certificate: the certificate DN of the client for external users; the Oracle Internet Directory DN for global users.
- For a proxy with user name: the Oracle Internet Directory DN if the client is an enterprise user; NULL if the client is a local database user.

Syntax

```
DVF.F$ENTERPRISE_IDENTITY ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check User Enterprise Identity',
  rule_expr => 'DVF.F$ENTERPRISE_IDENTITY NOT IN (''JSMITH'', ''TSMITH'')');
```

```
END;
/
```

F$IDENTIFICATION_TYPE Function

The F$IDENTIFICATION_TYPE function returns the way the schema of a user was created in the database. Specifically, it reflects the IDENTIFIED clause in the CREATE/ALTER USER syntax. The return type is VARCHAR2.

In the list that follows, the syntax used during schema creation is followed by the identification type returned:

- IDENTIFIED BY *password*: LOCAL
- IDENTIFIED EXTERNALLY: EXTERNAL
- IDENTIFIED GLOBALLY: GLOBAL SHARED
- IDENTIFIED GLOBALLY AS DN: GLOBAL PRIVATE

Syntax

```
DVF.F$IDENTIFICATION_TYPE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check User Schema Creation Type',
  rule_expr => 'DVF.F$IDENTIFICATION_TYPE = ''GLOBAL SHARED''');
END;
/
```

F$LANG Function

The F$LANG function returns the ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter, for the session of the user. The return type is VARCHAR2.

See *Oracle Database Globalization Support Guide* for a listing of supported languages for Oracle Database.

Syntax

```
DVF.F$LANG ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
```

```
  rule_name => 'Check ISO Abbreviated Language Name',
  rule_expr => 'DVF.F$LANG IN (''EN'', ''DE'', ''FR'')');
END;
/
```

F$LANGUAGE Function

The F$LANGUAGE function returns the language and territory currently used by a user session, along with the database character set. The return type is VARCHAR2.

The return type is in the following format:

language_territory.characterset

See *Oracle Database Globalization Support Guide* for a listing of supported languages and territories for Oracle Database.

Syntax

```
DVF.F$LANGUAGE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Session Language and Territory',
  rule_expr => 'DVF.F$LANGUAGE = ''AMERICAN_AMERICA.WE8ISO8859P1''');
END;
/
```

F$MACHINE Function

The F$MACHINE function returns the computer (host) name for the database client that established the database session. The return type is VARCHAR2.

Syntax

```
DVF.F$MACHINE ()
RETURN VARCHAR2;
```

Parameter

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client Computer Host Name',
  rule_expr => 'DVF.F$MACHINE NOT IN (''SHOBEEN'', ''SEBASTIAN'')');
END;
/
```

F$NETWORK_PROTOCOL Function

The F$NETWORK_PROTOCOL function returns the network protocol being used for communication, as specified in the PROTOCOL=*protocol* portion of the connect string. The return type is VARCHAR2.

Syntax

```
DVF.F$NETWORK_PROTOCOL ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Network Protocol',
  rule_expr => 'DVF.F$NETWORK_PROTOCOL = ''TCP''');
END;
/
```

F$PROXY_ENTERPRISE_IDENTITY Function

The F$PROXY_ENTERPRISE_IDENTITY function returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user. The return type is VARCHAR2.

Syntax

```
DVF.F$PROXY_ENTERPRISE_IDENTITY ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get OID DN of Enterprise User',
  rule_expr => 'DVF.F$PROXY_ENTERPRISE_IDENTITY = ''cn=Provisioning Admins''');
END;
/
```

F$PROXY_USER Function

The F$PROXY_USER function returns the name of a proxy user.

Syntax

```
DVF.PROXY_USER ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check Proxy Users',
   rule_expr => 'DVF.PROXY_USER NOT IN (''ECHICHESTER'', ''PFITCH'')');
END;/
```

F$SESSION_USER Function

The F$SESSION_USER function returns the database user name by which the current user is authenticated. This value remains the same throughout the session. The return type is VARCHAR2.

Syntax

```
DVF.F$SESSION_USER ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database User Name',
  rule_expr => 'DVF.F$SESSION_USER IN (''JSMITH'', ''TSMITH'')');
END;
/
```

19

Oracle Database Vault Secure Application Role APIs

The `DBMS_MACADM` and `DBMS_MACSEC_ROLES` PL/SQL packages manage Database Vault secure application roles.

- DBMS_MACADM Secure Application Role Procedures
  The `DBMS_MACADM` package creates, renames, assigns, unassigns, updates, and deletes Oracle Database Vault secure application roles.
- DBMS_MACSEC_ROLES Secure Application Role Procedure and Function
  The `DBMS_MACSEC_ROLES` package checks the authorization for users and sets Oracle Database Vault secure application roles.

Related Topics

- Configuring Secure Application Roles for Oracle Database Vault
  Secure application roles enable you to control how much access users have to an application.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

DBMS_MACADM Secure Application Role Procedures

The `DBMS_MACADM` package creates, renames, assigns, unassigns, updates, and deletes Oracle Database Vault secure application roles.

- ASSIGN_ROLE Procedure
  The `ASSIGN_ROLE` procedure assigns an Oracle Database Vault secure application role to a user.
- CREATE_ROLE Procedure
  The `CREATE_ROLE` procedure creates an Oracle Database Vault secure application role.
- DELETE_ROLE Procedure
  The `DELETE_ROLE` procedure deletes an Oracle Database Vault secure application role.
- RENAME_ROLE Procedure
  The `RENAME_ROLE` procedure renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.
- UPDATE_ROLE Procedure
  The `UPDATE_ROLE` procedure updates a Oracle Database Vault secure application role.
- UNASSIGN_ROLE Procedure
  The `ASSIGN_ROLE` procedure unassigns an Oracle Database Vault secure application role from a user.

ASSIGN_ROLE Procedure

The ASSIGN_ROLE procedure assigns an Oracle Database Vault secure application role to a user.

Syntax

```
DBMS_MACADM.ASSIGN_ROLE(
  role_name      IN VARCHAR2,
  assignee       IN VARCHAR2);
```

Parameters

Table 19-1 ASSIGN_ROLE Parameters

Parameter	Description
role_name	Role name, up to 128 characters, with no spaces. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View.
assignee	User to be assigned the secure application role To find existing database users in the current instance, query the DBA_USERS view, described in *Oracle Database Reference*.

Example

```
BEGIN
 DBMS_MACADM.ASSIGN_ROLE(
  role_name      => 'Sector2_APP_MGR',
  assignee       => 'PSMITH');
END;
/
```

CREATE_ROLE Procedure

The CREATE_ROLE procedure creates an Oracle Database Vault secure application role.

Syntax

```
DBMS_MACADM.CREATE_ROLE(
  role_name      IN VARCHAR2,
  enabled        IN VARCHAR2,
  rule_set_name  IN VARCHAR2);
```

Parameters

Table 19-2 CREATE_ROLE Parameters

Parameter	Description
role_name	Role name, up to 128 characters, with no spaces. In a multitenant environment, prepend the role name with C## or C##. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View.

Table 19-2 (Cont.) CREATE_ROLE Parameters

Parameter	Description
enabled	DBMS_MACUTL.G_YES (Yes) makes the role available for enabling; DBMS_MACUTL.G_NO (No) prevents the role from being enabled. The default is DBMS_MACUTL.G_YES.
rule_set_name	Name of rule set to determine whether this secure application can be enabled. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.

Example

```
BEGIN
 DBMS_MACADM.CREATE_ROLE(
  role_name      => 'Sector2_APP_MGR',
  enabled        => DBMS_MACUTL.G_YES,
  rule_set_name  => 'Check App2 Access');
END;
/
```

DELETE_ROLE Procedure

The DELETE_ROLE procedure deletes an Oracle Database Vault secure application role.

Syntax

```
DBMS_MACADM.DELETE_ROLE(
  role_name IN VARCHAR2);
```

Parameters

Table 19-3 DELETE_ROLE Parameter

Parameter	Description
role_name	Role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View.

Example

```
EXEC DBMS_MACADM.DELETE_ROLE('SECT2_APP_MGR');
```

RENAME_ROLE Procedure

The RENAME_ROLE procedure renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.

Syntax

```
DBMS_MACADM.RENAME_ROLE(
  role_name      IN VARCHAR2,
  new_role_name  IN VARCHAR2);
```

Parameters

Table 19-4 RENAME_ROLE Parameters

Parameter	Description
role_name	Current role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View.
new_role_name	Role name, up to 128 characters, with no spaces. Ensure that this name follows the standard Oracle naming conventions for role creation described in *Oracle Database SQL Language Reference*. In a multitenant environment, prepend the role name with c## or C##.

Example

```
BEGIN
 DBMS_MACADM.RENAME_ROLE(
  role_name      => 'SECT2_APP_MGR',
  new_role_name  => 'SECT2_SYSADMIN');
END;
/
```

UPDATE_ROLE Procedure

The UPDATE_ROLE procedure updates a Oracle Database Vault secure application role.

Syntax

```
DBMS_MACADM.UPDATE_ROLE(
  role_name      IN VARCHAR2,
  enabled        IN VARCHAR2,
  rule_set_name  IN VARCHAR2);
```

Parameters

Table 19-5 UPDATE_ROLE Parameters

Parameter	Description
role_name	Role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View.
enabled	DBMS_MACUTL.G_YES (Yes) makes the role available for enabling; DBMS_MACUTL.G_NO (No) prevents the role from being enabled. The default for enabled is the previously set value, which you can find by querying the DBA_DV_ROLE data dictionary view.
rule_set_name	Name of rule set to determine whether this secure application can be enabled. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_ROLE(
  role_name      => 'SECT2_SYSADMIN',
  enabled        => DBMS_MACUTL.G_YES,
  rule_set_name  => 'System Access Controls');
END;
/
```

UNASSIGN_ROLE Procedure

The ASSIGN_ROLE procedure unassigns an Oracle Database Vault secure application role from a user.

Syntax

```
DBMS_MACADM.UNASSIGN_ROLE(
  role_name      IN VARCHAR2,
  assignee       IN VARCHAR2);
```

Parameters

Table 19-6 UNASSIGN_ROLE Parameters

Parameter	Description
role_name	Role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View.
assignee	User who was assigned the Secure Application role To find existing database users in the current instance, query the DBA_USERS view, described in *Oracle Database Reference*.

Example

```
BEGIN
 DBMS_MACADM.UNASSIGN_ROLE(
  role_name      => 'Sector2_APP_MGR',
  assignee       => 'PSMITH');
END;
/
```

DBMS_MACSEC_ROLES Secure Application Role Procedure and Function

The DBMS_MACSEC_ROLES package checks the authorization for users and sets Oracle Database Vault secure application roles.

The DBMS_MACSEC_ROLES package is available to all users.

- CAN_SET_ROLE Function
  The CAN_SET_ROLE function checks if the user invoking the method is authorized to use an Oracle Database Vault secure application role.

- SET_ROLE Procedure
  The SET_ROLE procedure issues the SET ROLE PL/SQL statement for specified roles.

CAN_SET_ROLE Function

The CAN_SET_ROLE function checks if the user invoking the method is authorized to use an Oracle Database Vault secure application role.

The authorization is determined by checking the rule set associated with the role. The return type is BOOLEAN.

Syntax

```
DBMS_MACSEC_ROLES.CAN_SET_ROLE(
  p_role IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 19-7 CAN_SET_ROLE Parameter

Parameter	Description
p_role	Role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACSEC_ROLES.CAN_SET_ROLE('SECTOR2_APP_MGR')
  THEN DBMS_OUTPUT.PUT_LINE('''SECTOR2_APP_MGR'' can be enabled.');
 END IF;
END;
/
```

SET_ROLE Procedure

The SET_ROLE procedure issues the SET ROLE PL/SQL statement for specified roles.

This procedure includes both Oracle Database Vault secure application roles and regular Oracle Database roles in its checking process.

This procedure sets an Oracle Database Vault secure application role only if the rule set that is associated with the role evaluates to true. Before SET ROLE is issued, the CAN_SET_ROLE method is called to check the rule set associated with the role. Run-time rule set behavior such as auditing, failure processing, and event handling occur during this process.

The SET_ROLE procedure is available to the general database account population.

Syntax

```
DBMS_MACSEC_ROLES.SET_ROLE(
  p_role IN VARCHAR2);
```

Parameters

Table 19-8 SET_ROLE Parameter

Parameter	Description
p_role	Role names. You can enter multiple roles, separated by commas (,), including secure application roles and regular roles. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view, described in DBA_DV_ROLE View. To find all of the existing roles in the database, query the DBA_ROLES data dictionary view, described in *Oracle Database Reference*.

Example

```
EXEC DBMS_MACSEC_ROLES.SET_ROLE('SECTOR2_APP_MGR, APPS_MGR');
```

You can enter the name of the role in any case (for example, Sector2_APP_MGR).

20

Oracle Database Vault Oracle Label Security APIs

You can use the DBMS_MACADM PL/SQL package to manage Oracle Label Security labels and policies in Oracle Database Vault.

- CREATE_MAC_POLICY Procedure
  The CREATE_MAC_POLICY procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.
- CREATE_POLICY_LABEL Procedure
  The CREATE_POLICY_LABEL procedure labels an identity within an Oracle Label Security policy.
- DELETE_MAC_POLICY_CASCADE Procedure
  The DELETE_MAC_POLICY_CASCADE procedure deletes all Oracle Database Vault objects related to an Oracle Label Security policy.
- DELETE_POLICY_FACTOR Procedure
  The DELETE_POLICY_FACTOR procedure removes the factor from contributing to the Oracle Label Security label.
- DELETE_POLICY_LABEL Procedure
  The DELETE_POLICY_LABEL procedure removes the label from an identity within an Oracle Label Security policy.
- UPDATE_MAC_POLICY Procedure
  The UPDATE_MAC_POLICY procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Related Topics

- Integrating Oracle Database Vault with Other Oracle Products
  You can integrate Oracle Database Vault with other Oracle products, such as Oracle Enterprise User Security.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the DBMS_MACUTL PL/SQL package.

CREATE_MAC_POLICY Procedure

The CREATE_MAC_POLICY procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Syntax

```
DBMS_MACADM.CREATE_MAC_POLICY(
  policy_name  IN VARCHAR2,
  algorithm    IN VARCHAR2);
```

Parameters

Table 20-1 CREATE_MAC_POLICY Parameters

Parameter	Description
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DBA_DV_MAC_POLICY` view, described in DBA_DV_MAC_POLICY View.
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. Enter the code listed in Table 20-2 that corresponds to the merge algorithm you want. For example, enter `HUU` to if you want to select the Maximum Level/Union/Union merge algorithm.

Table 20-2 Oracle Label Security Merge Algorithm Codes

Code	Value
`HUU`	Maximum Level/Union/Union
`HIU`	Maximum Level/Intersection/Union
`HMU`	Maximum Level/Minus/Union
`HNU`	Maximum Level/Null/Union
`HUI`	Maximum Level/Union/Intersection
`HII`	Maximum Level/Intersection/Intersection
`HMI`	Maximum Level/Minus/Intersection
`HNI`	Maximum Level/Null/Intersection
`HUM`	Maximum Level/Union/Minus
`HIM`	Maximum Level/Intersection/Minus
`HMM`	Maximum Level/Minus/Minus
`HNM`	Maximum Level/Null/Minus
`HUN`	Maximum Level/Union/Null
`HIN`	Maximum Level/Intersection/Null
`HMN`	Maximum Level/Minus/Null
`HNN`	Maximum Level/Null/Null
`LUU`	Minimum Level/Union/Union
`LIU`	Minimum Level/Intersection/Union
`LMU`	Minimum Level/Minus/Union
`LNU`	Minimum Level/Null/Union
`LUI`	Minimum Level/Union/Intersection
`LII`	Minimum Level/Intersection/Intersection
`LMI`	Minimum Level/Minus/Intersection
`LNI`	Minimum Level/Null/Intersection
`LUM`	Minimum Level/Union/Minus

Table 20-2 (Cont.) Oracle Label Security Merge Algorithm Codes

Code	Value
LIM	Minimum Level/Intersection/Minus
LMM	Minimum Level/Minus/Minus
LNM	Minimum Level/Null/Minus
LUN	Minimum Level/Union/Null
LIN	Minimum Level/Intersection/Null
LMN	Minimum Level/Minus/Null
LNN	Minimum Level/Null/Null

Example

```
BEGIN
 DBMS_MACADM.CREATE_MAC_POLICY(
  policy_name  => 'Access Locations',
  algorithm    => 'HUU');
END;
/
```

CREATE_POLICY_LABEL Procedure

The CREATE_POLICY_LABEL procedure labels an identity within an Oracle Label Security policy.

Syntax

```
DBMS_MACADM.CREATE_POLICY_LABEL(
  identity_factor_name   IN VARCHAR2,
  identity_factor_value  IN VARCHAR2,
  policy_name            IN VARCHAR2,
  label                  IN VARCHAR2);
```

Parameters

Table 20-3 CREATE_POLICY_LABEL Parameters

Parameter	Description
identity_factor_name	Name of the factor being labeled. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in DBA_DV_FACTOR View. To find factors that are associated with Oracle Label Security policies, use DBA_DV_MAC_POLICY_FACTOR, described in DBA_DV_MAC_POLICY_FACTOR View.
identity_factor_value	Value of identity for the factor being labeled. To find the identities of existing factors in the current database instance, query the DBA_DV_IDENTITY view, described in DBA_DV_IDENTITY View.

Table 20-3 (Cont.) CREATE_POLICY_LABEL Parameters

Parameter	Description
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DBA_DV_MAC_POLICY view, described in DBA_DV_MAC_POLICY View.
label	Oracle Label Security label name. To find existing policy labels for factor identifiers, query the DBA_DV_POLICY_LABEL view, described in DBA_DV_POLICY_LABEL View.

Example

```
BEGIN
 DBMS_MACADM.CREATE_POLICY_LABEL(
  identity_factor_name   => 'App_Host_Name',
  identity_factor_value  => 'Sect2_Fin_Apps',
  policy_name            => 'Access Locations',
  label                  => 'Sensitive');
END;
/
```

DELETE_MAC_POLICY_CASCADE Procedure

The DELETE_MAC_POLICY_CASCADE procedure deletes all Oracle Database Vault objects related to an Oracle Label Security policy.

Syntax

```
DBMS_MACADM.DELETE_MAC_POLICY_CASCADE(
  policy_name  IN VARCHAR2);
```

Parameters

Table 20-4 DELETE_MAC_POLICY_CASCADE Parameter

Parameter	Description
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DBA_DV_MAC_POLICY view, described in DBA_DV_MAC_POLICY View.

Example

```
EXEC DBMS_MACADM.DELETE_MAC_POLICY_CASCADE('Access Locations');
```

DELETE_POLICY_FACTOR Procedure

The DELETE_POLICY_FACTOR procedure removes the factor from contributing to the Oracle Label Security label.

Syntax

```
DBMS_MACADM.DELETE_POLICY_FACTOR(
  policy_name  IN VARCHAR2,
  factor_name  IN VARCHAR2);
```

Parameters

Table 20-5 DELETE_POLICY_FACTOR Parameters

Parameter	Description
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DBA_DV_MAC_POLICY view, described in DBA_DV_MAC_POLICY View.
factor_name	Name of factor associated with the Oracle Label Security label. To find factors that are associated with Oracle Label Security policies, query DBA_DV_MAC_POLICY_FACTOR, described in DBA_DV_MAC_POLICY_FACTOR View.

Example

```
BEGIN
 DBMS_MACADM.DELETE_POLICY_FACTOR(
  policy_name  => 'Access Locations',
  factor_name  => 'App_Host_Name');
END;
/
```

DELETE_POLICY_LABEL Procedure

The DELETE_POLICY_LABEL procedure removes the label from an identity within an Oracle Label Security policy.

Syntax

```
DBMS_MACADM.DELETE_POLICY_LABEL(
  identity_factor_name   IN VARCHAR2,
  identity_factor_value  IN VARCHAR2,
  policy_name            IN VARCHAR2,
  label                  IN VARCHAR2);
```

Parameters

Table 20-6 DELETE_POLICY_LABEL Parameters

Parameter	Description
identity_factor_name	Name of the factor that was labeled. To find existing factors in the current database instance that are associated with Oracle Label Security policies, query DBA_DV_MAC_POLICY_FACTOR, described in DBA_DV_MAC_POLICY_FACTOR View.
identity_factor_value	Value of identity for the factor that was labeled. To find the identities of existing factors in the current database instance, query the DBA_DV_IDENTITY view, described in DBA_DV_IDENTITY View.
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DBA_DV_MAC_POLICY view, described in DBA_DV_MAC_POLICY View.
label	Oracle Label Security label name. To find existing policy labels for factor identifiers, query the DBA_DV_POLICY_LABEL view, described in DBA_DV_POLICY_LABEL View.

Example

```
BEGIN
 DBMS_MACADM.DELETE_POLICY_LABEL(
  identity_factor_name   => 'App_Host_Name',
  identity_factor_value  => 'Sect2_Fin_Apps',
  policy_name            => 'Access Locations',
  label                  => 'Sensitive');
END;
/
```

UPDATE_MAC_POLICY Procedure

The UPDATE_MAC_POLICY procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Syntax

```
DBMS_MACADM.UPDATE_MAC_POLICY(
  policy_name  IN VARCHAR2,
  algorithm    IN VARCHAR2);
```

Parameters

Table 20-7 UPDATE_MAC_POLICY

Parameter	Description
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DBA_DV_MAC_POLICY` view, described in DBA_DV_MAC_POLICY View.
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. See Table 20-2 for listing of the available algorithms.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_MAC_POLICY(
  policy_name  => 'Access Locations',
  algorithm    => 'LUI');
END;
/
```

21

Oracle Database Vault Utility APIs

Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

- DBMS_MACUTL Constants
  You can use a set of constants, available in the `DBMS_MACUTL` PL/SQL package.
- DBMS_MACUTL Package Procedures and Functions
  The `DBMS_MACUTL` PL/SQL package can perform tasks such as finding a time value or whether a user has the the appropriate privileges.

DBMS_MACUTL Constants

You can use a set of constants, available in the `DBMS_MACUTL` PL/SQL package.

- DBMS_MACUTL Listing of Constants
  The `DBMS_MACUTL` PL/SQL package provides constants (fields) to use with Oracle Database Vault PL/SQL packages.
- Example: Creating a Realm Using DBMS_MACUTL Constants
  Constants can be used to answer simple Yes or No settings when you create objects in Oracle Database Vault.
- Example: Creating a Rule Set Using DBMS_MACUTL Constants
  Constants can be used to set options such as the type of auditing used or fail options.
- Example: Creating a Factor Using DBMS_MACUTL Constants
  Constants can be used to set information specific to factors, such as identity or labeling.

DBMS_MACUTL Listing of Constants

The `DBMS_MACUTL` PL/SQL package provides constants (fields) to use with Oracle Database Vault PL/SQL packages.

Table 21-1 summarizes constant (that is, fields) descriptions for the `DBMS_MACUTL` package.

Many of these constants have equivalents in the Oracle Database Vault package. For example, the `enabled` parameter, which is available in several procedures, can accept either `Y` (for Yes) or the constant `G_YES`. Choosing one over the other is a matter of personal preference. They both have the same result.

Table 21-1 DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_ALL_OBJECT	VARCHAR2(1)	Used with the realm API `object_name` and `object_type` parameters as a wildcard to indicate all object names or all object types.
G_AUDIT_ALWAYS	NUMBER	Used with the factor API `audit_options` parameter to enable an audit.
G_AUDIT_OFF	NUMBER	Used with the factor API `audit_options` parameter to disable auditing.
G_AUDIT_ON_GET_ERROR	NUMBER	Used with the factor API `audit_options` parameter to audit if the expression specified in the `get_expr` parameter returns an error.
G_AUDIT_ON_GET_NULL	NUMBER	Used with the factor API `audit_options` parameter to audit if the expression in the `get_expr` field is null.
G_AUDIT_ON_TRUST_LEVEL_NEG	NUMBER	Used with the factor API `audit_options` parameter to audit if the trust level is negative.
G_AUDIT_ON_TRUST_LEVEL_NULL	NUMBER	Used with the factor API `audit_options` parameter to audit if no trust level exists.
G_AUDIT_ON_VALIDATE_ERROR	NUMBER	Used with the factor API `audit_options` parameter to audit if the validation function returns an error.
G_AUDIT_ON_VALIDATE_FALSE	NUMBER	Used with the factor API `audit_options` parameter to audit if validation function is false.
G_DISABLE	NUMBER	Used to disnable Oracle Database Vault policies and command rules
G_ENABLE	NUMBER	Used to enable Oracle Database Vault policies and command rules
G_EVAL_ON_ACCESS	NUMBER	Used with the factor API `eval_options` parameter to reevaluate the factor each time it is accessed.
G_EVAL_ON_SESSION	NUMBER	Used with the factor API `eval_options` parameter to evaluate the factor only once, when the user logs in to the session.
G_FAIL_SILENTLY	NUMBER	Used with the `fail_options` parameter to fail and show no error message.

Table 21-1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_FAIL_WITH_MESSAGE	NUMBER	Used with the fail_options parameter to fail and show an error message.
G_IDENTIFY_BY_CONSTANT	NUMBER	Used with the factor API identify_by parameter: Fixed value in PL/SQL expression defined in the get_expr parameter.
G_IDENTIFY_BY_CONTEXT	NUMBER	Used with the factor API identify_by parameter to indicate context.
G_IDENTIFY_BY_FACTOR	NUMBER	Used with the factor API identify_by parameter for subfactors through the factor_link$ table.
G_IDENTIFY_BY_METHOD	NUMBER	Used with the factor API identify_by parameter: Expression in get_expr field
G_IDENTIFY_BY_RULESET	NUMBER	Used with the factor API identify_by parameter: Expression and Rule Set with the factor_expr$ table
G_LABELED_BY_FACTORS	NUMBER	Used with the factor API labeled_by parameter to derive the label from subfactor and merge algorithm.
G_LABELED_BY_SELF	NUMBER	Used with the factor API labeled_by parameter to label the factor identities.
G_MAX_SESSION_LABEL	VARCHAR2(30)	This is the highest label a user could set based on the factors. It does not consider the label for a user.
G_MIN_POLICY_LABEL	VARCHAR2(30)	The label to which a factor with a null label defaults.
G_NO	VARCHAR2(1)	Used with the following APIs: • The factor API label_indicator parameter to indicate that a child factor linked to a parent factor does not contribute to the label of the parent factor in an Oracle Label Security integration. • Any API that uses the enabled parameter.
G_OLS_SESSION_LABEL	VARCHAR2(30)	The Oracle Label Security session label for a user at the time init_session is run.

Table 21-1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_PARTIAL	NUMBER	Sets the enforcement state of the realms and command rules under an Oracle Database Vault policy to be changed individually
G_REALM_AUDIT_FAIL	NUMBER	Used with the realm API `audit_options` parameter to audit when the realm is violated.
G_REALM_AUDIT_OFF	NUMBER	Used with the realm API `audit_options` parameter to disable auditing.
G_REALM_AUDIT_SUCCESS	NUMBER	Used with the realm API `audit_options parameter`: Audit on successful realm access
G_REALM_AUTH_OWNER	NUMBER	Used with the realm API `auth_options` parameter to set the realm authorization to Owner.
G_REALM_AUTH_PARTICIPANT	NUMBER	Used with the realm API `auth_options` parameter to set the realm authorization to Participant.
G_RULESET_AUDIT_FAIL	NUMBER	Used with the rule set API `audit_options` parameter to audit on rule set failure.
G_RULESET_AUDIT_OFF	NUMBER	Used with the rule set API `audit_options` parameter to disable auditing.
G_RULESET_AUDIT_SUCCESS	NUMBER	Used with the rule set API `audit_options` parameter to audit on rule set success.
G_RULESET_EVAL_ALL	NUMBER	Used with the rule set API `eval_options` parameter to enable the rule set to succeed if all rules evaluate to true.
G_RULESET_EVAL_ANY	NUMBER	Used with the rule set API `eval_options` parameter to succeed if any of the rules evaluate to true.
G_RULESET_FAIL_SHOW	NUMBER	Used with the rule set API `fail_options` parameter to show an error message if the rule set fails.
G_RULESET_FAIL_SILENT	NUMBER	Used with the rule set API `fail_options` parameter to not show an error message if the rule set fails.
G_RULESET_HANDLER_FAIL	NUMBER	Used with the rule set API `handler_options` parameter to call a handler (specified by the `handler` parameter) if the rule set fails.

Table 21-1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_RULESET_HANDLER_OFF	NUMBER	Used with the rule set API handler_options parameter to disable calls to a handler or if no handler is used.
G_RULESET_HANDLER_SUCCESS	NUMBER	Used with the rule set API handler_options parameter to call a handler if the rule set succeeds.
G_SIMULATION	NUMBER	Used to set the enforcement state of a policy to simulation mode. This mode does not raise errors for realm or command rule violations. Instead, an error is logged in a designated log table with sufficient information relevant to the error (for example, users or SQL command.)
G_USER_POLICY_LABEL	VARCHAR2(30)	This is what Oracle Label Security has decided the user's label should be set to after factoring in the preceding values.
G_YES	VARCHAR2(1)	Used with the following APIs: • The factor API label_indicator parameter to indicate that a child factor linked to a parent factor contributes to the label of the parent factor in an Oracle Label Security integration. • Any API that uses the enabled parameter.

Example: Creating a Realm Using DBMS_MACUTL Constants

Constants can be used to answer simple Yes or No settings when you create objects in Oracle Database Vault.

Example 21-1 shows how to use the G_YES and G_REALM_AUDIT_FAIL DBMS_MACUTL constants when creating a realm.

Example 21-1 Creating a Realm Using DBMS_MACUTL Constants

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Performance Statistics Realm',
  description   => 'Realm to measure performance',
  enabled       =>  DBMS_MACUTL.G_YES,
  audit_options =>  DBMS_MACUTL.G_REALM_AUDIT_FAIL);
END;
/
```

Example: Creating a Rule Set Using DBMS_MACUTL Constants

Constants can be used to set options such as the type of auditing used or fail options.

Example 21-2 shows how to use several DBMS_MACUTL constants when creating a rule set.

Example 21-2 Creating a Rule Set Using DBMS_MACUTL Constants

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'Rule Set Limit_DBA_Access has failed.',
  fail_code        => 20000,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler          => 'dbavowner.email_alert');
END;
/
```

Example: Creating a Factor Using DBMS_MACUTL Constants

Constants can be used to set information specific to factors, such as identity or labeling.

Example 21-3 shows how to use constants when creating a factor.

Example 21-3 Creating a Factor Using DBMS_MACUTL Constants

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'DB_access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_FACTOR,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_SESSION,
  audit_options     => DBMS_MACUTL.G_AUDIT_ALWAYS,
  fail_options      => DBMS_MACUTL.G_FAIL_SILENTLY);
END;
/
```

DBMS_MACUTL Package Procedures and Functions

The DBMS_MACUTL PL/SQL package can perform tasks such as finding a time value or whether a user has the the appropriate privileges.

- CHECK_DVSYS_DML_ALLOWED Procedure
  The CHECK_DVSYS_DML_ALLOWED procedure checks if a user can issue Data Modification Language (DML) commands to access the DVSYS objects.

- GET_CODE_VALUE Function
  The GET_CODE_VALUE function finds the value for a code within a code group, and then returns a VARCHAR2 value.
- GET_SECOND Function
  The GET_SECOND function returns the seconds in Oracle SS (seconds) format (00–59), and then returns a NUMBER value.
- GET_MINUTE Function
  The GET_MINUTE function returns the minute in Oracle MI (minute) format (00–59), in a NUMBER value.
- GET_HOUR Function
  The GET_HOUR function returns the hour in Oracle HH24 (hour) format (00–23), in a NUMBER value.
- GET_DAY Function
  The GET_DAY function returns the day in Oracle DD (day) format (01–31), in a NUMBER value.
- GET_MONTH Function
  The GET_MONTH function returns the month in Oracle MM (month) format (01–12), in a NUMBER value.
- GET_YEAR Function
  The GET_YEAR function returns the year in Oracle YYYY (year) format (0001–9999), in a NUMBER value.
- IS_ALPHA Function
  The IS_ALPHA function returns a BOOLEAN value indicating if a character is alphabetic.
- IS_DIGIT Function
  The IS_DIGIT function checks returns a BOOLEAN value indicating if a character is numeric.
- IS_DVSYS_OWNER Function
  The IS_DVSYS_OWNER function returns a BOOLEAN value indicating if a user is authorized to manage the Oracle Database Vault configuration.
- IS_OLS_INSTALLED Function
  The IS_OLS_INSTALLED function returns a BOOLEAN value indicating if Oracle Label Security is installed.
- IS_OLS_INSTALLED_VARCHAR Function
  The IS_OLS_INSTALLED_VARCHAR function returns a BOOLEAN value indicating if Oracle Label Security is installed.
- USER_HAS_OBJECT_PRIVILEGE Function
  The USER_HAS_OBJECT_PRIVILEGE function returns a BOOLEAN value indicating if user or role can access an object through a single specified object privilege grant.
- USER_HAS_ROLE Function
  The USER_HAS_ROLE function returns a BOOLEAN value indicating if a user has a role privilege, directly or indirectly (through another role).
- USER_HAS_ROLE_VARCHAR Function
  The USER_HAS_ROLE_VARCHAR function returns a VARCHAR2 value indicating if a user has a role privilege, directly or indirectly (through another role).

- USER_HAS_SYSTEM_PRIVILEGE Function
  The USER_HAS_SYSTEM_PRIVILEGE function returns a BOOLEAN value indicating if a user has a system privilege, directly or indirectly (through a role).

CHECK_DVSYS_DML_ALLOWED Procedure

The CHECK_DVSYS_DML_ALLOWED procedure checks if a user can issue Data Modification Language (DML) commands to access the DVSYS objects.

Syntax

```
DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED(
  p_user IN VARCHAR2 DEFAULT USER);
```

Parameter

Table 21-2 CHECK_DVSYS_DML_ALLOWED Parameter

Parameter	Description
p_user	User to check. To find existing users in the current database instance, query the following views: • DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See DBA_DV_REALM_AUTH View. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See DBA_DV_ROLE View.

Example

User SYSTEM fails the check:

```
EXEC DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED('system');

ERROR at line 1:
ORA-47920: Authorization failed for user system to perform this operation
ORA-06512: at "DBMS_MACUTL", line 23
ORA-06512: at "DBMS_MACUTL", line 372
ORA-06512: at "DBMS_MACUTL", line 508
ORA-06512: at "DBMS_MACUTL", line 572
ORA-06512: at line 1
```

User leo_dvowner, who has the DV_OWNER role, passes the check:

```
EXEC DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED('leo_dvowner');

PL/SQL procedure successfully completed.
```

GET_CODE_VALUE Function

The GET_CODE_VALUE function finds the value for a code within a code group, and then returns a VARCHAR2 value.

Syntax

```
DBMS_MACUTL.GET_CODE_VALUE(
  p_code_group IN VARCHAR2,
  p_code       IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 21-3 GET_CODE_VALUE Parameters

Parameter	Description
p_code_group	Code group (for example, AUDIT_EVENTS or BOOLEAN). To find available code groups in the current database instance, query the DBA_DV_CODE view, described in DBA_DV_CODE View.
p_code	ID of the code. This ID is listed when you run the DBA_DV_CODE view.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Label Algorithm for Maximum Level/Union/Null',
  rule_expr => 'DBMS_MACUTL.GET_CODE_VALUE(''LABEL_ALG'', ''HUN'') = ''Union''');
END;
/
```

GET_SECOND Function

The GET_SECOND function returns the seconds in Oracle SS (seconds) format (00–59), and then returns a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_SECOND(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 21-4 GET_SECOND Parameter

Parameter	Description
p_date	Date in SS format (for example, 59). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  seconds number;
BEGIN
  seconds := DBMS_MACUTL.GET_SECOND(TO_DATE('03-APR-2009 6:56 PM',
  'dd-mon-yyyy hh:mi PM'));
  DBMS_OUTPUT.PUT_LINE('Seconds: '||seconds);
END;
/
```

This example, which uses a fixed date and time, returns the following:

```
Seconds: 56
```

GET_MINUTE Function

The GET_MINUTE function returns the minute in Oracle MI (minute) format (00–59), in a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_MINUTE(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 21-5 GET_MINUTE Parameter

Parameter	Description
p_date	Date in MI format (for example, 30, as in 2:30). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  minute number;
```

```
BEGIN
  minute := DBMS_MACUTL.GET_MINUTE(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Minute: '||minute);
END;
/
```

Output similar to the following appears:

```
Minute: 17
```

GET_HOUR Function

The GET_HOUR function returns the hour in Oracle HH24 (hour) format (00–23), in a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_HOUR(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 21-6 GET_HOUR Parameter

Parameter	Description
p_date	Date in HH24 format (for example, 14 for 2:00 p.m.) If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  hours number;
BEGIN
  hours := DBMS_MACUTL.GET_HOUR(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Hour: '||hours);
END;
/
```

Output similar to the following appears:

```
Hour: 12
```

GET_DAY Function

The GET_DAY function returns the day in Oracle DD (day) format (01–31), in a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_DAY(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 21-7 GET_DAY Parameter

Parameter	Description
p_date	Date in DD format (for example, 01 for the first day of the month). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  day number;
BEGIN
  day := DBMS_MACUTL.GET_DAY(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Day: '||day);
END;
/
```

Output similar to the following appears:

```
Day: 3
```

GET_MONTH Function

The GET_MONTH function returns the month in Oracle MM (month) format (01–12), in a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_MONTH(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 21-8 GET_MONTH Parameter

Parameter	Description
p_date	Date in MM format (for example, 08 for the month of August). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  month number;
BEGIN
  month := DBMS_MACUTL.GET_MONTH(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Month: '||month);
END;
/
```

Output similar to the following appears:

```
Month: 4
```

GET_YEAR Function

The GET_YEAR function returns the year in Oracle YYYY (year) format (0001–9999), in a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_YEAR(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 21-9 GET_YEAR Parameter

Parameter	Description
p_date	Date in YYYY format (for example, 1984). If you do not specify a date, then Oracle Database Vault uses the SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  year number;
BEGIN
  year := DBMS_MACUTL.GET_YEAR(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Year: '||year);
END;
/
```

IS_ALPHA Function

The IS_ALPHA function returns a BOOLEAN value indicating if a character is alphabetic.

IS_ALPHA returns TRUE if the character is alphabetic.

Syntax

```
DBMS_MACUTL.IS_ALPHA(
  c IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 21-10 IS_ALPHA Parameter

Parameter	Description
c	String with one character

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_ALPHA('z')
  THEN DBMS_OUTPUT.PUT_LINE('The alphabetic character was found');
 ELSE
  DBMS_OUTPUT.PUT_LINE('No alphabetic characters today.');
 END IF;
END;
/
```

IS_DIGIT Function

The `IS_DIGIT` function checks returns a `BOOLEAN` value indicating if a character is numeric.

`IS_DIGIT` returns `TRUE` if the character is a digit.

Syntax

```
DBMS_MACUTL.IS_DIGIT(
  c IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 21-11 IS_DIGIT Parameter

Parameter	Description
c	String with one character

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_DIGIT('7')
  THEN DBMS_OUTPUT.PUT_LINE('The numeric character was found');
 ELSE
  DBMS_OUTPUT.PUT_LINE('No numeric characters today.');
 END IF;
```

```
END;
/
```

IS_DVSYS_OWNER Function

The IS_DVSYS_OWNER function returns a BOOLEAN value indicating if a user is authorized to manage the Oracle Database Vault configuration.

IS_DVSYS_OWNER returns TRUE if the user is authorized.

Syntax

```
DBMS_MACUTL.IS_DVSYS_OWNER(
  p_user IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameter

Table 21-12 IS_DVSYS_OWNER Parameter

Parameter	Description
p_user	User to check. To find existing users, query the following views: • DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See DBA_DV_REALM_AUTH View. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See DBA_DV_ROLE View.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_DVSYS_OWNER('PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('PSMITH is authorized to manage Database Vault.');
 ELSE
  DBMS_OUTPUT.PUT_LINE('PSMITH is not authorized to manage Database Vault.');
 END IF;
END;
/
```

IS_OLS_INSTALLED Function

The IS_OLS_INSTALLED function returns a BOOLEAN value indicating if Oracle Label Security is installed.

If Oracle Label Security is installed, IS_OLS_INSTALLED returns TRUE.

Syntax

```
DBMS_MACUTL.IS_OLS_INSTALLED()
RETURN BOOLEAN;
```

Parameters

None

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_OLS_INSTALLED()
  THEN DBMS_OUTPUT.PUT_LINE('OLS is installed');
 ELSE
  DBMS_OUTPUT.PUT_LINE('OLS is not installed');
 END IF;
END;
/
```

IS_OLS_INSTALLED_VARCHAR Function

The `IS_OLS_INSTALLED_VARCHAR` function returns a `BOOLEAN` value indicating if Oracle Label Security is installed.

If Oracle Label Security is installed, then `IS_OLS_INSTALLED_VARCHAR` returns `Y`.

Syntax

```
DBMS_MACUTL.IS_OLS_INSTALLED_VARCHAR()
RETURN VARCHAR2;
```

Parameters

None

Example

See IS_OLS_INSTALLED Function for an example.

USER_HAS_OBJECT_PRIVILEGE Function

The `USER_HAS_OBJECT_PRIVILEGE` function returns a `BOOLEAN` value indicating if user or role can access an object through a single specified object privilege grant.

If the user or role has the object privilege, then `USER_HAS_OBJECT_PRIVILEGE` returns `TRUE`.

Syntax

```
DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE(
  p_user          VARCHAR2,
  p_object_owner  VARCHAR2,
  p_object_name   VARCHAR2,
  p_privilege     VARCHAR2)
RETURNS BOOLEAN;
```

Parameters

Table 21-13 USER_HAS_OBJECT_PRIVILEGE Parameters

Parameter	Description
p_user	User or role to check. To find existing users, query they following views: • DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. • DBA_ROLES: Finds available roles in the current database instance. See *Oracle Database Reference*. • DVA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See DBA_DV_REALM_AUTH View. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See DBA_DV_ROLE View.
p_object_owner	Object owner, such as a schema. To find the available users, query they DBA_USERS view, described in *Oracle Database Reference*. To find the authorization of a particular user, query they DVA_DV_REALM_AUTH view.
p_object_name	Object name, such as a table within the schema specified in the p_object_owner parameter. To find the available objects, query the ALL_OBJECTS view, described in *Oracle Database Reference*. To find objects that are secured by existing realms, query they DBA_DV_REALM_OBJECT view.
p_privilege	Object privilege, such as, UPDATE. To find privileges for a database account excluding PUBLIC privileges, query they DBA_DV_USER_PRIVS view. To find all privileges for a database account, query the DBA_DV_USER_PRIVS_ALL. view

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE(
   'SECTOR2_APP_MGR', 'OE', 'ORDERS', 'UPDATE')
  THEN DBMS_OUTPUT.PUT_LINE('SECTOR2_APP_MGR has the UPDATE privilege for the
OE.ORDERS table');
   ELSE
  DBMS_OUTPUT.PUT_LINE('SECTOR2_APP_MGR does not have the UPDATE privilege for the
OE.ORDERS table.');
 END IF;
END;
/
```

USER_HAS_ROLE Function

The USER_HAS_ROLE function returns a BOOLEAN value indicating if a user has a role privilege, directly or indirectly (through another role).

If the user has a role privilege, then USER_HAS_ROLE returns TRUE.

Syntax

```
DBMS_MACUTL.USER_HAS_ROLE(
  p_role IN VARCHAR2,
  p_user IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameters

Table 21-14 USER_HAS_ROLE Parameters

Parameter	Description
p_role	Role privilege to check. To find existing roles, query the following views: • DBA_ROLES: Finds available roles in the current database instance. See *Oracle Database Reference*. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See DBA_DV_REALM_AUTH View. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See DBA_DV_ROLE View.
p_user	User to check. To find existing users, query the following views: • DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See DBA_DV_REALM_AUTH View.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.USER_HAS_ROLE('SECTOR2_APP_MGR', 'PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('User PSMITH has the SECTOR2_APP_MGR role');
   ELSE
  DBMS_OUTPUT.PUT_LINE('User PSMITH does not have the SECTOR2_APP_MGR role.');
 END IF;
END;
/
```

USER_HAS_ROLE_VARCHAR Function

The USER_HAS_ROLE_VARCHAR function returns a VARCHAR2 value indicating if a user has a role privilege, directly or indirectly (through another role).

If the user has the role privilege specified, then USER_HAS_ROLE_VARCHAR returns Y.

Syntax

```
DBMS_MACUTL.USER_HAS_ROLE_VARCHAR(
  p_role IN VARCHAR2,
  p_user IN VARCHAR2 DEFAULT USER)
RETURN VARCHAR2;
```

Parameters

Table 21-15 USER_HAS_ROLE_VARCHAR Parameters

Parameter	Description
p_role	Role to check. To find existing roles, query the following views: • DBA_ROLES: Finds available roles in the current database instance. See *Oracle Database Reference*. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See DBA_DV_REALM_AUTH View. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See DBA_DV_ROLE View.
p_user	User to check. To find existing users, query the following views: • DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See DBA_DV_REALM_AUTH View.

USER_HAS_SYSTEM_PRIVILEGE Function

The USER_HAS_SYSTEM_PRIVILEGE function returns a BOOLEAN value indicating if a user has a system privilege, directly or indirectly (through a role).

If the user has the system privilege specified, then USER_HAS_SYSTEM_PRIVILEGE returns TRUE.

Syntax

```
DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE(
  p_privilege  IN VARCHAR2,
  p_user       IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameters

Table 21-16 USER_HAS_SYSTEM_PRIVILEGE Parameters

Parameter	Description
p_privilege	System privilege to check for. To find privileges for a database account excluding PUBLIC privileges, query the DBA_DV_USER_PRIVS view, described in DBA_DV_USER_PRIVS View. To find all privileges for a database account, use DBA_DV_USER_PRIVS_ALL, described in DBA_DV_USER_PRIVS_ALL View.
p_user	User to check. To find existing users, query the following views: • DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See DBA_DV_REALM_AUTH View.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE('EXECUTE', 'PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('User PSMITH has the EXECUTE ANY PRIVILEGE privilege.');
   ELSE
  DBMS_OUTPUT.PUT_LINE('User PSMITH does not have the EXECUTE ANY PRIVILEGE
privilege.');
 END IF;
END;
/
```

22

Oracle Database Vault General Administrative APIs

The DBMS_MACADM PL/SQL package and the CONFIGURE_DV standalone procedure enable you to you perform general maintenance tasks.

- DBMS_MACADM General System Maintenance Procedures
  The DBMS_MACADM PL/SQL package general system maintenance procedures perform tasks such as authorizing users or adding new language to Oracle Database Vault.
- CONFIGURE_DV General System Maintenance Procedure
  The CONFIGURE_DV procedure configures the initial two Oracle Database user accounts, which are granted the DV_OWNER and DV_ACCTMGR roles, respectively.

DBMS_MACADM General System Maintenance Procedures

The DBMS_MACADM PL/SQL package general system maintenance procedures perform tasks such as authorizing users or adding new language to Oracle Database Vault.

- ADD_NLS_DATA Procedure
  The ADD_NLS_DATA procedure adds a new language to Oracle Database Vault.
- AUTHORIZE_DATAPUMP_USER Procedure
  The AUTHORIZE_DATAPUMP_USER procedure authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled.
- AUTHORIZE_DBCAPTURE Procedure
  The AUTHORIZE_DBCAPTURE procedure grants a user authorization to perform Oracle Database Replay workload capture operations.
- AUTHORIZE_DBREPLAY Procedure
  The AUTHORIZE_DBREPLAY procedure grants a user authorization to perform Oracle Database Replay workload replay operations.
- AUTHORIZE_DDL Procedure
  The AUTHORIZE_DDL procedure grants a user authorization to execute Data Definition Language (DDL) statements on the specified schema.
- AUTHORIZE_MAINTENANCE_USER Procedure
  The AUTHORIZE_MAINTENANCE_USER procedure grants a user authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.
- AUTHORIZE_PREPROCESSOR Procedure
  The AUTHORIZE_PREPROCESSOR procedure grants a user authorization to execute preprocessor programs through external tables.
- AUTHORIZE_PROXY_USER Procedure
  The AUTHORIZE_PROXY_USER procedure grants a proxy user authorization to proxy other user accounts, as long as the proxy user has database authorization.

- AUTHORIZE_SCHEDULER_USER Procedure
  The AUTHORIZE_SCHEDULER_USER procedure grants a user authorization to schedule database jobs when Oracle Database Vault is enabled.
- AUTHORIZE_TTS_USER Procedure
  The AUTHORIZE_TTS_USER procedure authorizes a user to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled.
- UNAUTHORIZE_DATAPUMP_USER Procedure
  The UNAUTHORIZE_DATAPUMP_USER procedure revokes the authorization that was granted by the AUTHORIZE_DATAPUMP_USER procedure.
- UNAUTHORIZE_DBCAPTURE Procedure
  The UNAUTHORIZE_DBCAPTURE procedure revokes authorization from users to perform Oracle Database Replay workload capture operations.
- UNAUTHORIZE_DBREPLAY Procedure
  The UNAUTHORIZE_DBREPLAY procedure revokes authorization from users to perform Oracle Database Replay workload replay operations.
- UNAUTHORIZE_DDL Procedure
  The UNAUTHORIZE_DDL procedure revokes authorization from a user who was granted authorization to execute DDL statements through the DBMS_MACDM.AUTHORIZE_DDL procedure.
- UNAUTHORIZE_MAINTENANCE_USER Procedure
  The UNAUTHORIZE_MAINTENANCE_USER procedure revokes privileges from users who have been granted authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.
- UNAUTHORIZE_PREPROCESSOR Procedure
  The UNAUTHORIZE_PREPROCESSOR procedure revokes authorization from a user to execute preprocessor programs through external tables.
- UNAUTHORIZE_PROXY_USER Procedure
  The UNAUTHORIZE_PROXY_USER procedure revokes authorization from a user who was granted proxy authorization from the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure.
- UNAUTHORIZE_SCHEDULER_USER Procedure
  The UNAUTHORIZE_SCHEDULER_USER procedure revokes the authorization that was granted by the AUTHORIZE_SCHEDULER_USER procedure.
- UNAUTHORIZE_TTS_USER Procedure
  The UNAUTHORIZE_TTS_USER procedure removes from authorization users who had previously been granted the authorization to perform Oracle Data Pump transportable tablespace operations.
- DISABLE_DV Procedure
  The DISABLE_DV procedure disables Oracle Database Vault.
- DISABLE_DV_DICTIONARY_ACCTS Procedure
  The DISABLE_DV_DICTIONARY_ACCTS procedure prevents any user from logging into the database as the DVSYS or DVF schema user.
- DISABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The DISABLE_DV_PATCH_ADMIN_AUDIT procedure disables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.

- DISABLE_ORADEBUG Procedure
  The DISABLE_ORADEBUG procedure disables the use of the ORADEBUG utility in an Oracle Database Vault environment.
- ENABLE_DV Procedure
  The ENABLE_DV procedure enables Oracle Database Vault and Oracle Label Security.
- ENABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The ENABLE_DV_PATCH_ADMIN_AUDIT procedure enables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.
- ENABLE_DV_DICTIONARY_ACCTS Procedure
  The ENABLE_DV_DICTIONARY_ACCTS procedure enables users to log into the database as the DVSYS or DVF user.
- ENABLE_ORADEBUG Procedure
  The ENABLE_ORADEBUG procedure enables the use of the ORADEBUG utility in an Oracle Database Vault environment.

ADD_NLS_DATA Procedure

The ADD_NLS_DATA procedure adds a new language to Oracle Database Vault.

Syntax

```
DBMS_MACADM.ADD_NLS_DATA(
  language      IN VARCHAR );
```

Parameters

Table 22-1 ADD_NLS_DATA

Parameter	Description
language	Enter one of the following settings. (This parameter is case insensitive.) • ENGLISH • GERMAN • SPANISH • FRENCH • ITALIAN • JAPANESE • KOREAN • BRAZILIAN PORTUGUESE • SIMPLIFIED CHINESE • TRADITIONAL CHINESE

Examples

```
EXEC DBMS_MACADM.ADD_NLS_DATA('french');
```

AUTHORIZE_DATAPUMP_USER Procedure

The AUTHORIZE_DATAPUMP_USER procedure authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled.

It applies to both the expdp and impdp utilities.

See Authorizing Users or Roles for Oracle Data Pump Regular Operations in Database Vault for full usage information, including the levels of additional authorization the user must have to use Oracle Data Pump in an Oracle Database Vault environment.

Syntax

```
DBMS_MACADM.AUTHORIZE_DATAPUMP_USER(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2 DEFAULT NULL,
  table_name     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 22-2 AUTHORIZE_DATAPUMP_USER

Parameter	Description
user_name	Name of the Oracle Data Pump user to whom you want to grant authorization. To find a list of users who have privileges to use Oracle Data Pump (that is, the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles), query the DBA_ROLE_PRIVS data dictionary view as follows: `SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE LIKE '%FULL%'`
schema_name	Name of the database schema that the Oracle Data Pump user must export or import. If you omit this parameter, then the user is granted global authorization to export and import any schema in the database. In this case, ensure the user has been granted the DV_OWNER role.
table_name	Name of the table within the schema specified by the schema_name parameter. If you omit this parameter, then the user you specified can export and import all tables within the schema specified by the schema_name parameter.

Examples

```
EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR');

EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR');

EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR', 'EMPLOYEES');
```

AUTHORIZE_DBCAPTURE Procedure

The AUTHORIZE_DBCAPTURE procedure grants a user authorization to perform Oracle Database Replay workload capture operations.

To find information about users who have been granted this authorization, query the DBA_DV_DBCAPTURE_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_DBCAPTURE(
  uname      IN VARCHAR2);
```

Parameters

Table 22-3 AUTHORIZE_DBCAPTURE

Parameter	Description
uname	Name of the user to whom you want to grant Database Replay workload capture authorization

Example 22-1 Example

```
EXEC DBMS_MACADM.AUTHORIZE_DBCAPTURE('PFITCH');
```

AUTHORIZE_DBREPLAY Procedure

The AUTHORIZE_DBREPLAY procedure grants a user authorization to perform Oracle Database Replay workload replay operations.

To find information about users who have been granted this authorization, query the DBA_DV_DBREPLAY_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_DBREPLAY(
  uname      IN VARCHAR2);
```

Parameters

Table 22-4 AUTHORIZE_DBREPLAY

Parameter	Description
uname	Name of the user to whom you want to grant Database Replay workload replay authorization

Example 22-2 Example

```
EXEC DBMS_MACADM.AUTHORIZE_DBREPLAY('PFITCH');
```

AUTHORIZE_DDL Procedure

The AUTHORIZE_DDL procedure grants a user authorization to execute Data Definition Language (DDL) statements on the specified schema.

To find information about users who have been granted this authorization, query the DBA_DV_DDL_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_DDL(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2);
```

Parameters

Table 22-5 AUTHORIZE_DDL

Parameter	Description
user_name	Name of the user to whom you want to grant DDL authorization.
schema_name	Name of the database schema in which the user wants to perform the DDL statements. Enter % to specify all schemas.

Examples

The following example enables user psmith to execute DDL statements in any schema:

```
EXEC DBMS_MACADM.AUTHORIZE_DDL('psmith', '%');
```

This example enables user psmith to execute DDL statements in the HR schema only.

```
EXEC DBMS_MACADM.AUTHORIZE_DDL('psmith', 'HR');
```

AUTHORIZE_MAINTENANCE_USER Procedure

The AUTHORIZE_MAINTENANCE_USER procedure grants a user authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

To find information about users who have been granted this authorization, query the DBA_DV_MAINTENANCE_AUTH view.

Syntax

```
DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER(
 uname         IN VARCHAR2,
 sname         IN VARCHAR2 DEFAULT NULL,
 objname       IN VARCHAR2 DEFAULT NULL,
 objtype       IN VARCHAR2 DEFAULT NULL,
 action        IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 22-6 AUTHORIZE_MAINTENANCE_USER

Parameter	Description
uname	Name of the user to whom you want to grant authorization
sname	Name of the database schema for which the maintenance operations are to be performed. Enter % to specify all schemas.
objname	Name of the object (such as the name of a table) in the schema that is specified in the sname parameter for which maintenance operations are to be performed
objtype	Type of the objname object, such as table, index, tablespace, and so on
action	Maintenance action. Enter ilm for Information Lifecycle Management

Example

The following example enables user psmith to have Database Vault authorization to manage ILM features for the HR.EMPLOYEES table:

```
BEGIN
 DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER (
  uname       => 'psmith',
  sname       => 'HR',
  objname     => 'EMPLOYEES',
  objtype     => 'TABLE',
  action      => 'ILM');
END;
/
```

Related Topics

- Using Information Lifecycle Management with Oracle Database Vault
  Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.

AUTHORIZE_PREPROCESSOR Procedure

The AUTHORIZE_PREPROCESSOR procedure grants a user authorization to execute preprocessor programs through external tables.

To find information about users who have been granted this authorization, query the DBA_DV_PREPROCESSOR_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_PREPROCESSOR(
  uname      IN VARCHAR2);
```

Parameters

Table 22-7 AUTHORIZE_PREPROCESSOR

Parameter	Description
uname	Name of the user to whom you want to grant authorization to execute preprocessor programs through external tables

Example 22-3 Example

```
EXEC DBMS_MACADM.AUTHORIZE_PREPROCESSOR('PFITCH');
```

Related Topics

- Executing Preprocessor Programs with Oracle Database Vault
  Users who execute preprocessor programs through external tables must have Oracle Database Vault-specific authorization.
- DBA_DV_PREPROCESSOR_AUTH View
  The DBA_DV_PREPROCESSOR_AUTH data dictionary view shows users who have been granted authorization to execute preprocessor programs through external tables.

AUTHORIZE_PROXY_USER Procedure

The AUTHORIZE_PROXY_USER procedure grants a proxy user authorization to proxy other user accounts, as long as the proxy user has database authorization.

For example, the CREATE SESSION privilege is a valid database authorization.

AUTHORIZE_PROXY_USER does not control whether a particular user can connect as a proxy of another user. That part is controlled by GRANT CONNECT THROUGH, which can be issued only by the a user who has the DV_ACCTMGR role. Instead, AUTHORIZE_PROXY_USER controls whether the proxy user is allowed to assume all the Database Vault authorizations that the target user has. For example, suppose that the proxy user hr_proxy_user successfully connects as user HR. Now being HR, hr_proxy_user can access all the objects to which HR has access. However, if the target objects are Database Vault protected and HR is authorized to access it, hr_proxy_user can access the objects if and only if hr_proxy_user is proxy-authorized for HR. If hr_proxy_user is not proxy-authorized for HR, then even after connecting as HR, hr_proxy_user cannot access the Database Vault-protected objects for which HR is authorized.

To find information about users who have been granted authorization using AUTHORIZE_PROXY_USER, query the DBA_DV_PROXY_AUTH view.

Syntax

```
DBMS_MACADM.AUTHORIZE_PROXY_USER(
  proxy_user   IN VARCHAR2,
  user_name    IN VARCHAR2);
```

Parameters

Table 22-8 AUTHORIZE_PROXY_USER

Parameter	Description
`proxy_user`	Name of the proxy user.
`user_name`	Name of the database user who will be proxied by the `proxy_user` user. Enter `%` to specify all users.

Examples

The following example enables proxy user `preston` to proxy all users:

```
EXEC DBMS_MACADM.AUTHORIZE_PROXY_USER('preston', '%');
```

This example enables proxy user `preston` to proxy database user `dkent` only.

```
EXEC DBMS_MACADM.AUTHORIZE_PROXY_USER('preston', 'dkent');
```

AUTHORIZE_SCHEDULER_USER Procedure

The `AUTHORIZE_SCHEDULER_USER` procedure grants a user authorization to schedule database jobs when Oracle Database Vault is enabled.

This authorization applies to anyone who has privileges to schedule database jobs. These privileges include any of the following: `CREATE JOB`, `CREATE ANY JOB`, `CREATE EXTERNAL JOB`, `EXECUTE ANY PROGRAM`, `EXECUTE ANY CLASS`, `MANAGE SCHEDULER`. See Using Oracle Scheduler with Oracle Database Vault full usage information, including the levels of authorization the user must have to schedule database jobs in an Oracle Database Vault environment.

Syntax

```
DBMS_MACADM.AUTHORIZE_SCHEDULER_USER(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 22-9 AUTHORIZE_SCHEDULER_USER

Parameter	Description
`user_name`	Name of the user to whom you want to grant authorization. To find a list of users who have privileges to schedule jobs, query the `DBA_SYS_PRIVS` data dictionary view. See Step 2 in Granting a Job Scheduling Administrator Authorization for Database Vault.
`schema_name`	Name of the database schema for which a job will be scheduled. If you omit this parameter, then the user is granted global authorization to schedule a job for any schema in the database.

Examples

The following example authorizes the user `JOB_MGR` to run a job under any schema.

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR');
```

This example authorizes user JOB_MGR to run a job under the HR schema only.

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

AUTHORIZE_TTS_USER Procedure

The AUTHORIZE_TTS_USER procedure authorizes a user to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled.

It applies to both the EXPDP and IMPDP utilities.

Authorizing Users or Roles for Oracle Data Pump Regular Operations in Database Vault describes full usage information, including the levels of additional authorization the user must have to use Oracle Data Pump to conduct transportable operations in an Oracle Database Vault environment.

Syntax

```
DBMS_MACADM.AUTHORIZE_TTS_USER(
  uname      IN VARCHAR2,
  tsname     IN VARCHAR2);
```

Parameters

Table 22-10 AUTHORIZE_TTS_USER

Parameter	Description
uname	Name of the user who you want to authorize to perform Oracle Data Pump transportable tablespace operations. To find a list of users and their current privileges, query the DBA_SYS_PRIVS data dictionary view.
tsname	Name of the tablespace in which the uname user is to perform the transportable tablespace operation. To find a list of tablespaces, query the DBA_TABLESPACES data dictionary view.

Example

```
EXEC DBMS_MACADM.AUTHORIZE_TTS_USER('PSMITH', 'HR_TS');
```

UNAUTHORIZE_DATAPUMP_USER Procedure

The UNAUTHORIZE_DATAPUMP_USER procedure revokes the authorization that was granted by the AUTHORIZE_DATAPUMP_USER procedure.

When you run this procedure, ensure that its settings correspond exactly to the equivalent AUTHORIZE_DATAPUMP_USER procedure.

For example, the following two procedures will work because the parameters are consistent:

```
EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR');

EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('DP_MGR');
```

However, because the parameters in the following procedures are not consistent, the UNAUTHORIZE_DATAPUMP_USER procedure will not work:

```
EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('JSMITH');

EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('JSMITH', 'HR');
```

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2 DEFAULT NULL,
  table_name     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 22-11 UNAUTHORIZE_DATAPUMP_USER

Parameter	Description
user_name	Name of the Oracle Data Pump user from whom you want to revoke authorization. To find a list of users and authorizations from the AUTHORIZE_DATAPUMP_USER procedure, query the DBA_DV_DATAPUMP_AUTH data dictionary view as follows: `SELECT * FROM DBA_DV_DATAPUMP_AUTH;`
schema_name	Name of the database schema that the Oracle Data Pump user is authorized to export or import.
table_name	Name of the table within the schema specified by the schema name parameter.

Examples

```
EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('JSMITH');

EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('JSMITH', 'HR');

EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('JSMITH', 'HR', 'SALARY');
```

UNAUTHORIZE_DBCAPTURE Procedure

The UNAUTHORIZE_DBCAPTURE procedure revokes authorization from users to perform Oracle Database Replay workload capture operations.

To find information about users who have been granted this authorization, query the DBA_DV_DBCAPTURE_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DBCAPTURE(
  uname      IN VARCHAR2);
```

Parameters

Table 22-12 UNAUTHORIZE_DBCAPTURE

Parameter	Description
uname	Name of the user from whom you want to revoke Database Replay workload capture authorization

Example 22-4 Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_DBCAPTURE('PFITCH');
```

UNAUTHORIZE_DBREPLAY Procedure

The UNAUTHORIZE_DBREPLAY procedure revokes authorization from users to perform Oracle Database Replay workload replay operations.

To find information about users who have been granted this authorization, query the DBA_DV_DBREPLAY_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DBREPLAY(
  uname       IN VARCHAR2);
```

Parameters

Table 22-13 UNAUTHORIZE_DBREPLAY

Parameter	Description
uname	Name of the user from whom you want to revoke Database Replay workload replay authorization

Example 22-5 Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_DBREPLAY('PFITCH');
```

UNAUTHORIZE_DDL Procedure

The UNAUTHORIZE_DDL procedure revokes authorization from a user who was granted authorization to execute DDL statements through the DBMS_MACDM.AUTHORIZE_DDL procedure.

To find information about users who have been granted this authorization, query the DBA_DV_DDL_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DDL(
  user_name       IN VARCHAR2,
  schema_name     IN VARCHAR2);
```

Parameters

Table 22-14 UNAUTHORIZE_DDL

Parameter	Description
user_name	Name of the user from whom you want to revoke DDL authorization.
schema_name	Name of the database schema in which the user wants to perform the DDL statements. Enter % specify all schemas.

Examples

The following example revokes DDL statement execution authorization from user psmith for all schemas:

```
EXEC DBMS_MACADM.UNAUTHORIZE_DDL('psmith', '%');
```

This example revokes DDL statement execution authorization from user psmith for the HR schema only.

```
EXEC DBMS_MACADM.UNAUTHORIZE_DDL('psmith', 'HR');
```

UNAUTHORIZE_MAINTENANCE_USER Procedure

The UNAUTHORIZE_MAINTENANCE_USER procedure revokes privileges from users who have been granted authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

To find information about the settings for the ILM authorization, query the DBA_DV_MAINTENANCE_AUTH view.

When you run this procedure, ensure that its settings correspond exactly to the equivalent AUTHORIZE_MAINTENANCE_USER procedure.

For example, the following two procedures will work because the parameter settings correspond:

```
EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER('psmith', 'OE', 'ORDERS', 'TABLE',
'ILM');
EXEC DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER('psmith', 'OE', 'ORDERS', 'TABLE',
'ILM');
```

However, these two statements will fail because the settings do not correspond:

```
EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER('psmith', 'OE', 'ORDERS', 'TABLE',
'ILM');

EXEC DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER('psmith', '%', '%', '%', 'ILM');
```

Syntax

```
DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER(
 uname         IN VARCHAR2,
 sname         IN VARCHAR2 DEFAULT NULL,
 objname       IN VARCHAR2 DEFAULT NULL,
 objtype       IN VARCHAR2 DEFAULT NULL,
 action        IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 22-15 UNAUTHORIZE_MAINTENANCE_USER

Parameter	Description
uname	Name of the user to whom you want to revoke authorization
sname	Name of the database schema for which the maintenance operations are performed. Enter % to specify all schemas.
objname	Name of the object (such as the name of a table) in the schema that is specified in the sname parameter for which maintenance operations are performed
objtype	Type of the objname object, such as table, index, tablespace, and so on
action	Maintenance action. Enter ilm for Information Lifecycle Management

Example

The following example revokes privileges from Database Vault user psmith so that she can no longer perform ILM operations in any HR schema objects:

```
BEGIN
 DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER (
  uname       => 'psmith',
  sname       => 'HR',
  objname     => 'EMPLOYEES',
  objtype     => 'TABLE',
  action      => 'ILM');
END;
/
```

Related Topics

- Using Information Lifecycle Management with Oracle Database Vault
  Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.

UNAUTHORIZE_PREPROCESSOR Procedure

The UNAUTHORIZE_PREPROCESSOR procedure revokes authorization from a user to execute preprocessor programs through external tables.

To find information about users who have been granted this authorization, query the DBA_DV_PREPROCESSOR_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR(
  uname      IN VARCHAR2);
```

Parameters

Table 22-16 UNAUTHORIZE_PREPROCESSOR

Parameter	Description
uname	Name of the user from whom you want to revoke authorization to execute preprocessor programs through external tables

Example 22-6 Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR('PFITCH');
```

Related Topics

- Executing Preprocessor Programs with Oracle Database Vault
  Users who execute preprocessor programs through external tables must have Oracle Database Vault-specific authorization.
- DBA_DV_PREPROCESSOR_AUTH View
  The DBA_DV_PREPROCESSOR_AUTH data dictionary view shows users who have been granted authorization to execute preprocessor programs through external tables.

UNAUTHORIZE_PROXY_USER Procedure

The UNAUTHORIZE_PROXY_USER procedure revokes authorization from a user who was granted proxy authorization from the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_PROXY_USER(
  proxy_user   IN VARCHAR2,
  user_name    IN VARCHAR2);
```

Parameters

Table 22-17 UNAUTHORIZE_PROXY_USER

Parameter	Description
proxy_user	Name of the proxy user.
user_name	Name of the database user who was proxied by the proxy_user user. Enter % to specify all users.

Examples

The following example revokes proxy authorization from user preston for proxying all users:

```
DBMS_MACADM.UNAUTHORIZE_PROXY_USER('preston', '%');
```

This example revokes proxy authorization from user preston for proxying database user psmith only.

```
EXEC DBMS_MACADM.UNAUTHORIZE_PROXY_USER('preston', 'psmith');
```

UNAUTHORIZE_SCHEDULER_USER Procedure

The UNAUTHORIZE_SCHEDULER_USER procedure revokes the authorization that was granted by the AUTHORIZE_SCHEDULER_USER procedure.

When you run this procedure, ensure that its settings correspond exactly to the equivalent AUTHORIZE_SCHEDULER_USER procedure. For example, the following two procedures will work because the parameters are consistent:

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR');

EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR');
```

However, because the parameters in the following procedures are not consistent, the UNAUTHORIZE_SCHEDULER_USER procedure will not work:

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR');

EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

Syntax

```
DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 22-18 UNAUTHORIZE_SCHEDULER_USER

Parameter	Description
user_name	Name of the job scheduling user from whom you want to revoke authorization. To find a list of users and authorizations from the AUTHORIZE_SCHEDULER_USER procedure, query the DBA_DV_JOB_AUTH data dictionary view as follows: `SELECT * FROM DBA_DV_JOB_AUTH;`
schema_name	Name of the database schema for which the user is authorized to schedule jobs.

Examples

```
EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR');

EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

UNAUTHORIZE_TTS_USER Procedure

The UNAUTHORIZE_TTS_USER procedure removes from authorization users who had previously been granted the authorization to perform Oracle Data Pump transportable tablespace operations.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_TTS_USER
  uname      IN VARCHAR2,
  tsname     IN VARCHAR2);
```

Parameters

Table 22-19 UNAUTHORIZE_TTS_USER

Parameter	Description
uname	Name of the user who you want to remove from being authorized to perform Oracle Data Pump transportable tablespace operations. To find a list of users and their current privileges, query the DBA_SYS_PRIVS data dictionary view.
tsname	Name of the tablespace that is used in the transportable tablespace operation. To find a list of tablespaces, query the DBA_TABLESPACES data dictionary view.

Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_TTS_USER('PSMITH', 'HR_TS');
```

DISABLE_DV Procedure

The DISABLE_DV procedure disables Oracle Database Vault.

After you run this procedure, you must restart the database.

Syntax

```
DBMS_MACADM.DISABLE_DV;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.DISABLE_DV;
```

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

DISABLE_DV_DICTIONARY_ACCTS Procedure

The DISABLE_DV_DICTIONARY_ACCTS procedure prevents any user from logging into the database as the DVSYS or DVF schema user.

By default these two accounts are locked. Only a user who has been granted the DV_OWNER role can execute this procedure. To find the status of whether users can log into DVSYS and DVF, query the DBA_DV_DICTIONARY_ACCTS data dictionary view. For stronger security, run this procedure to better protect the DVSYS and DVF schemas. The disablement takes place immediately, so you do not need to restart the database after running this procedure.

Syntax

```
DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS;
```

Related Topics

- Archiving and Purging the Oracle Database Vault Audit Trail
  If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.

DISABLE_DV_PATCH_ADMIN_AUDIT Procedure

The DISABLE_DV_PATCH_ADMIN_AUDIT procedure disables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.

This procedure disables the successful actions of this user, not the failed actions. You should run this procedure after the DV_PATCH_ADMIN user has completed database patch operation. To find if auditing is enabled or not, query the DBA_DV_PATCH_AUDIT data dictionary view.

Syntax

```
DBMS_MACADM.DISABLE_DV_PATCH_ADMIN_AUDIT;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.DISABLE_DV_PATCH_ADMIN_AUDIT;
```

Related Topics

- DV_PATCH_ADMIN Database Vault Database Patch Role
  The DV_PATCH_ADMIN role is used for patching operations.

- ENABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The ENABLE_DV_PATCH_ADMIN_AUDIT procedure enables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.

DISABLE_ORADEBUG Procedure

The DISABLE_ORADEBUG procedure disables the use of the ORADEBUG utility in an Oracle Database Vault environment.

The disablement takes place immediately, so you do not need to restart the database after running this procedure. To find the status of whether the ORADEBUG utility is available in Database Vault, query the DVYS.DBA_DV_ORADEBUG data dictionary view.

Syntax

```
DBMS_MACADM.DISABLE_ORADEBUG;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.DISABLE_ORADEBUG;
```

Related Topics

- Using the ORADEBUG Utility with Oracle Database Vault
  The ORADEBUG utility is used primarily by Oracle Support to diagnose problems that may arise with an Oracle database.

ENABLE_DV Procedure

The ENABLE_DV procedure enables Oracle Database Vault and Oracle Label Security.

After you run this procedure, you must restart the database.

Syntax

```
DBMS_MACADM.ENABLE_DV(
 strict_mode  IN VARCHAR2 DEFAULT);
```

Parameters

Table 22-20 ENABLE_DV

Parameter	Description
strict_mode	In a multitenant environment, specifies one of the following modes: • n specifies regular mode, which allows the PDBs to be Database Vault enabled or disabled. (Default) • y specifies strict mode, which puts the PDBs that have not been Database Vault-enabled in restricted mode, until you enable Database Vault in them and then restart the PDB. To apply this setting to all PDBs in the multitenant environment, run the DBMS_MACADM.ENABLE_DV procedure in the CDB root. To apply it to all PDBs in an application container, run the procedure in the application root. In a non-multitenant environment, omit this parameter.

Examples

The following example enables Oracle Database Vault in regular mode.

```
EXEC DBMS_MACADM.ENABLE_DV;
```

This example enables Oracle Database Vault in strict mode in a multitenant environment.

```
EXEC DBMS_MACADM.ENABLE_DV (strict_mode => 'y');
```

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

ENABLE_DV_PATCH_ADMIN_AUDIT Procedure

The ENABLE_DV_PATCH_ADMIN_AUDIT procedure enables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.

This procedure is designed to audit these users' actions during a patch upgrade. To find if this auditing is enabled or not, query the DBA_DV_PATCH_AUDIT data dictionary view.

Syntax

```
DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT;
```

Related Topics

- DV_PATCH_ADMIN Database Vault Database Patch Role
  The DV_PATCH_ADMIN role is used for patching operations.
- DISABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The DISABLE_DV_PATCH_ADMIN_AUDIT procedure disables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.

ENABLE_DV_DICTIONARY_ACCTS Procedure

The ENABLE_DV_DICTIONARY_ACCTS procedure enables users to log into the database as the DVSYS or DVF user.

By default, the DVSYS and DVF accounts are locked.

Only a user who has been granted the DV_OWNER role can execute this procedure. To find the status of whether users can log into DVSYS and DVF, query the DBA_DV_DICTIONARY_ACCTS data dictionary view. For stronger security, only run this procedure when you need to better protect the DVSYS and DVF schemas. The enablement takes place immediately, so you do not need to restart the database after running this procedure.

Syntax

```
DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS;
```

Related Topics

- Archiving and Purging the Oracle Database Vault Audit Trail
  If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.

ENABLE_ORADEBUG Procedure

The ENABLE_ORADEBUG procedure enables the use of the ORADEBUG utility in an Oracle Database Vault environment.

The enablement takes place immediately, so you do not need to restart the database after running this procedure. To find the status of whether the ORADEBUG utility is available in Database Vault, query the DVYS.DBA_DV_ORADEBUG data dictionary view.

Syntax

```
DBMS_MACADM.ENABLE_ORADEBUG;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.ENABLE_ORADEBUG;
```

Related Topics

- Using the ORADEBUG Utility with Oracle Database Vault
  The ORADEBUG utility is used primarily by Oracle Support to diagnose problems that may arise with an Oracle database.

CONFIGURE_DV General System Maintenance Procedure

The CONFIGURE_DV procedure configures the initial two Oracle Database user accounts, which are granted the DV_OWNER and DV_ACCTMGR roles, respectively.

You can check the status of this configuration by querying the DBA_DV_STATUS data dictionary view. Before you run the CONFIGURE_DV procedure, you must create the two user accounts and grant them the CREATE SESSION privilege. The accounts can be either local or common. If you create common user accounts, then the Database Vault roles that are granted to these users apply to the current pluggable database (PDB) only. You then refer to these user accounts for the CONFIGURE_DV procedure.

The CONFIGURE_DV procedure resides in the SYS schema. Oracle provides a synonym, DVSYS.CONFIGURE_DV, so that any existing Oracle Database Vault configuration scripts that you may have created in previous releases will continue to work in this release.

You only can run the CONFIGURE_DV procedure once, when you are ready to register Oracle Database Vault with an Oracle database. After you run this procedure, you must run utlrp.sql script and then DBMS_MACADM.ENABLE_DV to complete the registration process. Oracle strongly recommends that for better security, you use the two accounts you create here as back-up accounts and then create additional accounts for every day use. See Backup Oracle Database Vault Accounts for guidance.

When you run the CONFIGURE_DV procedure, it checks the DVSYS schema for problems such as missing tables or packages. If it finds problems, then it raises an ORA-47500 Database Vault cannot be configured error. If this happens, then you must deinstall and then reinstall Oracle Database Vault.

Together, the CONFIGURE_DV and DBMS_MACADM.ENABLE_DV procedures, and the and utlrp.sql script, are designed to be a command-line alternative to using Oracle Database Configuration Assistant (DBCA) to register Oracle Database Vault with an Oracle database.

You must run the CONFIGURE_DV procedure as user SYS. Registering Oracle Database Vault with an Oracle Database in a Multitenant Environment describes the process that you would use.

Syntax

```
CONFIGURE_DV
  dvowner_uname         IN VARCHAR2,
  dvacctmgr_uname       IN VARCHAR2;
```

Parameters

Table 22-21 CONFIGURE_DV

Parameter	Description
dvowner_uname	Name of the user who will be the Database Vault Owner. This user will be granted the DV_OWNER role.
dvacctmgr_uname	Name of the user who will be the Database Vault Account Manager. This user will be granted the DV_ACCTMGR role. If you omit this setting, the user specified by the dvowner_uname parameter is made the Database Vault Account Manager and granted the DV_ACCTMGR role.

Example

```
CREATE USER dbv_owner IDENTIFIED BY password CONTAINER = CURRENT;
CREATE USER dbv_acctmgr IDENTIFIED BY password CONTAINER = CURRENT;
GRANT CREATE SESSION TO dbv_owner, dbv_acctmgr;

BEGIN
 CONFIGURE_DV (
   dvowner_uname         => 'dbv_owner',
   dvacctmgr_uname       => 'dbv_acctmgr');
 END;
/
```

Related Topics

- Deinstalling Oracle Database Vault
  You can remove Oracle Database Vault from an Oracle Database installation, for both to both single-instance and Oracle RAC installations.
- Reinstalling Oracle Database Vault
  You can reinstall Oracle Database Vault by using Database Configuration Assistant and afterward, register Database Vault.

23
Oracle Database Vault Policy APIs

You can use the `DBMS_MACADM` PL/SQL package to manage Oracle Database Vault policies.

Only users who have been granted the `DV_OWNER` or `DV_ADMIN` role can use these procedures.

- ADD_CMD_RULE_TO_POLICY Procedure
  The `ADD_COMMAND_RULE_TO_POLICY` procedure enables you to add an existing command rule to an Oracle Database Vault policy.
- ADD_OWNER_TO_POLICY Procedure
  The `ADD_OWNER_TO_POLICY` procedure enables you to add an existing database user to an Oracle Database Vault policy as an owner.
- ADD_REALM_TO_POLICY Procedure
  The `ADD_REALM_TO_POLICY` procedure enables you to add an existing realm to an Oracle Database Vault policy.
- CREATE_POLICY Procedure
  The `CREATE_POLICY` procedure enables you to create an Oracle Database Vault policy.
- DELETE_CMD_RULE_FROM_POLICY Procedure
  The `DELETE_CMD_RULE_FROM_POLICY` procedure enables you to remove an existing command rule from an Oracle Database Vault policy.
- DELETE_OWNER_FROM_POLICY Procedure
  The `DELETE_OWNER_FROM_POLICY` procedure enables you to remove an owner from an Oracle Database Vault policy.
- DELETE_REALM_FROM_POLICY Procedure
  The `DELETE_REALM_FROM_POLICY` procedure enables you to remove an existing realm from an Oracle Database Vault policy.
- DROP_POLICY Procedure
  The `DROP_POLICY` procedure enables you to drop an existing Oracle Database Vault policy.
- RENAME_POLICY Procedure
  The `UPDATE_POLICY_DESCRIPTION` procedure enables you to rename an existing Oracle Database Vault policy.
- UPDATE_POLICY_DESCRIPTION Procedure
  The `UPDATE_POLICY_DESCRIPTION` procedure enables you to update the `description` field in an Oracle Database Vault policy.
- UPDATE_POLICY_STATE Procedure
  The `UPDATE_POLICY_STATE` procedure enables you to update the `policy_state` field in an Oracle Database Vault policy.

Related Topics

- Configuring Oracle Database Vault Policies
  You can use Oracle Database Vault policies to implement frequently used realm and command rule settings.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

ADD_CMD_RULE_TO_POLICY Procedure

The `ADD_COMMAND_RULE_TO_POLICY` procedure enables you to add an existing command rule to an Oracle Database Vault policy.

You can add a command rule to a policy when the command rule is in any state. For example, you can add a disabled command rule to an enabled policy. In this case, the disabled command rule will automatically become enabled when it is added to the policy. A command rules can be added to only one policy. In other words, you cannot assign the same command rule to multiple policies.

Syntax

```
DBMS_MACADM.ADD_CMD_RULE_TO_POLICY(
 policy_name    IN VARCHAR2,
 command        IN VARCHAR2,
 object_owner   IN VARCHAR2,
 object_name    IN VARCHAR2,
 clause_name    IN VARCHAR2 DEFAULT,
 parameter_name IN VARCHAR2 DEFAULT,
 event_name     IN VARCHAR2 DEFAULT,
 component_name IN VARCHAR2 DEFAULT,
 action_name    IN VARCHAR2 DEFAULT,
 scope          IN NUMBER DEFAULT);
```

Parameters

Table 23-1 ADD_CMD_RULE_TO_POLICY Parameters

Parameter	Description
`policy_name`	Policy name. To find existing Database Vault policies in the current database instance, query the `DBA_DV_POLICY` view, described in DBA_DV_POLICY View.
`command`	Command rule name To find existing Database Vault command rules in the current database instance, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View.
`object_owner`	Database schema to which the command rule applies To find existing object owners for this command rule, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View
`object_name`	Object to be protected by the command rule To find existing objects for this command rule, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View

Table 23-1 (Cont.) ADD_CMD_RULE_TO_POLICY Parameters

Parameter	Description
clause_name	For ALTER SYSTEM and ALTER SESSION command rules, a clause from the SQL statement that was used to create the command rule To find existing clauses for this command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View
parameter_name	For ALTER SYSTEM and ALTER SESSION command rules, a parameter from the clause_name parameter. To find existing parameters for this command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View
event_name	For ALTER SYSTEM and ALTER SESSION command rules, an event that the command rule defines To find existing event names for this command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View
component_name	A component of the event_name setting To find existing component names for this command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View
action_name	An action of the component_name setting. To find existing action names for this command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule applies to all the PDBs

Example

The following example shows how to add a common command rule to a Database Vault policy. This command rule is in the application root of a multitenant environment, so the user running this procedure must be in the application root or the CDB root. Any rules or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.ADD_CMD_RULE_TO_POLICY(
  policy_name    => 'HR_DV_Policy',
  command        => 'ALTER SESSION',
  object_owner   => '%',
  object_name    => '%',
  clause_name    => 'PARALLEL DDL',
  parameter_name => '',
  event_name     => '',
  action_name    => '',
  scope          => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

ADD_OWNER_TO_POLICY Procedure

The ADD_OWNER_TO_POLICY procedure enables you to add an existing database user to an Oracle Database Vault policy as an owner.

When you add an owner to an enabled policy, the change takes place immediately. There is no limit to the number of users that you add to the policy.

Syntax

```
DBMS_MACADM.ADD_OWNER_TO_POLICY(
  policy_name    IN VARCHAR2,
  owner_name     IN VARCHAR2);
```

Parameters

Table 23-2 ADD_OWNER_TO_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view, described in DBA_DV_POLICY View.
owner_name	User name. To find existing database users (not roles) in the current instance, query the DBA_USERS view, described in *Oracle Database Reference*. To find existing policy owners, query the DBA_DV_POLICY_OWNER view, described in DBA_DV_POLICY_OWNER View.

Example

```
BEGIN
 DBMS_MACADM.ADD_OWNER_TO_POLICY(
  policy_name     => 'HR_DV_Policy',
  owner_name      => 'PSMITH');
END;
/
```

ADD_REALM_TO_POLICY Procedure

The ADD_REALM_TO_POLICY procedure enables you to add an existing realm to an Oracle Database Vault policy.

You can add a disabled realm to an enabled policy. In this case, the realm automatically becomes enabled when it is added. A realm can be added to only one policy. In other words, you cannot assign the same realm to multiple policies.

Syntax

```
DBMS_MACADM.ADD_REALM_TO_POLICY(
  policy_name    IN VARCHAR2,
  realm_name     IN VARCHAR2);
```

Parameters

Table 23-3 ADD_REALM_TO_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view, described in DBA_DV_POLICY View.
realm_name	Realm name. To find existing Database Vault realms in the current database instance, query the DV_REALM view, described in DVSYS.DV$REALM View.

Example

```
BEGIN
 DBMS_MACADM.ADD_REALM_TO_POLICY(
  policy_name    => 'HR_DV_Policy',
  realm_name     => 'HR realm');
END;
/
```

CREATE_POLICY Procedure

The CREATE_POLICY procedure enables you to create an Oracle Database Vault policy.

After you create the policy, you must add at least one realm and one command rule to the policy. Optionally, you can set these realms and command rules to be enforced individually or use the enforcement that the policy uses.

An owner for the policy is not required, but if you do not assign an owner to the policy, a user who has been granted the DV_OWNER or DV_ADMIN role must administer the policy.

After you create the policy, use the following procedures to complete the policy definition:

- ADD_REALM_TO_POLICY adds realms to the policy.
- ADD_CMD_RULE_TO_POLICY adds command rules to the policy.
- ADD_OWNER_TO_POLICY enables the specified database users to manage the policy.

Syntax

```
DBMS_MACADM.CREATE_POLICY(
 policy_name  IN VARCHAR2,
 description  IN VARCHAR2 DEFAULT,
 policy_state IN NUMBER,
 pl_sql_stack IN BOOLEAN DEFAULT);
```

Parameters

Table 23-4 CREATE_POLICY Parameters

Parameter	Description
`policy_name`	Policy name, up to 128 characters in mixed case To find existing policies in the current database instance, query the `DBA_DV_POLICY` view, described in DBA_DV_POLICY View.
`description`	Description of the purpose of the policy, up to 4000 characters in mixed-case.
`policy_state`	Specifies how the policy is enabled. Possible values are: • `DBMS_MACADM.G_ENABLED` (1), which enables the policy after you create it • `DBMS_MACADM.G_DISABLED` (0), which disables the policy after you create it • `DBMS_MACADM.G_SIMULATION` (2), which sets the policy to simulation mode. In simulation mode, any violations to realms or command rules used in the policy are logged in a designated log table with sufficient information to describe the error, such as the user name or SQL statement used. • `DBMS_MACADM.G_PARTIAL` (3), which sets the policy to partial mode. In partial mode, the enforcement state of realms or command rules associated with the policy can be changed individually. See About Simulation Mode for more information about simulation mode
`pl_sql_stack`	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter `TRUE` to record the PL/SQL stack, `FALSE` to not record.

Example

The following example creates a policy that uses the partial state and enables the capture of the PL/SQL stack. Later on, when a realm or a command rule is added to this policy, their enforcement state will be able to be changed individually.

```
BEGIN
 DBMS_MACADM.CREATE_POLICY(
  policy_name  => 'HR Database Vault Policy',
  description  => 'Policy to protect the HR schema',
  policy_state => DBMS_MACADM.G_ENABLED,
  pl_sql_stack => TRUE);
END;
/
```

DELETE_CMD_RULE_FROM_POLICY Procedure

The `DELETE_CMD_RULE_FROM_POLICY` procedure enables you to remove an existing command rule from an Oracle Database Vault policy.

You can remove command rules from a policy anytime regardless of the state of the policy. When a command rule is removed from a policy, the state of command rule remains the same. That is, if the policy is enabled, and a command rule is removed from the policy, then the command rule will be still enabled after you have removed it from the policy.

Syntax

```
DBMS_MACADM.DELETE_CMD_RULE_FROM_POLICY(
 policy_name    IN VARCHAR2,
 command        IN VARCHAR2,
 object_owner   IN VARCHAR2,
 object_name    IN VARCHAR2,
 clause_name    IN VARCHAR2 DEFAULT,
 parameter_name IN VARCHAR2 DEFAULT,
 event_name     IN VARCHAR2 DEFAULT,
 component_name IN VARCHAR2 DEFAULT,
 action_name    IN VARCHAR2 DEFAULT,
 scope          IN NUMBER DEFAULT);
```

Parameters

Table 23-5 DELETE_CMD_RULE_FROM_POLICY Parameters

Parameter	Description
`policy_name`	Policy name. To find existing Database Vault policies in the current database instance, query the `DBA_DV_POLICY` view, described in DBA_DV_POLICY View.
`command`	Command rule name To find existing Database Vault command rules in the current database instance, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View.
`object_owner`	Database schema to which the command rule applies To find existing object owners for this command rule, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View
`object_name`	Object to be protected by the command rule To find existing objects for this command rule, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View
`clause_name`	For `ALTER SYSTEM` and `ALTER SESSION` command rules, a clause from the SQL statement that was used to create the command rule To find existing clauses for this command rule, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View
`parameter_name`	For `ALTER SYSTEM` and `ALTER SESSION` command rules, a parameter from the `clause_name` parameter. To find existing parameters for this command rule, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View
`event_name`	For `ALTER SYSTEM` and `ALTER SESSION` command rules, an event that the command rule defines To find existing event names for this command rule, query the `DBA_DV_COMMAND_RULE` view, described in DBA_DV_COMMAND_RULE View

Table 23-5 (Cont.) DELETE_CMD_RULE_FROM_POLICY Parameters

Parameter	Description
component_name	A component of the event_name setting To find existing component names for this command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View
action_name	An action of the component_name setting. To find existing action names for this command rule, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View
scope	For a multitenant environment, determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Example

The following example shows how to delete a common command rule from a Database Vault policy. This command rule is in the application root of a multitenant environment, so the user running this procedure must be in the CDB root.

```
BEGIN
 DBMS_MACADM.DELETE_CMD_RULE_FROM_POLICY(
  policy_name    => 'HR_DV_Policy',
  command        => 'ALTER SESSION',
  object_owner   => '%',
  object_name    => '%',
  clause_name    => 'END SESSION',
  parameter_name => 'KILL SESSION',
  event_name     => '',
  action_name    => '',
  scope          => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

DELETE_OWNER_FROM_POLICY Procedure

The DELETE_OWNER_FROM_POLICY procedure enables you to remove an owner from an Oracle Database Vault policy.

You can remove owners from policies any time, regardless of the state (enabled or disabled) of the policy. The change takes effect immediately.

Syntax

```
DBMS_MACADM.DELETE_OWNER_FROM_POLICY(
  policy_name    IN VARCHAR2,
  owner_name     IN VARCHAR2);
```

Parameters

Table 23-6 DELETE_OWNER_FROM_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view, described in DBA_DV_POLICY View.
owner_name	User name. To find existing policy owners in the current instance, query the DBA_DV_POLICY_OWNER view, described in DBA_DV_POLICY_OWNER View.

Example

```
BEGIN
 DBMS_MACADM.DELETE_OWNER_FROM_POLICY(
  policy_name    => 'HR_DV_Policy',
  owner_name     => 'PSMITH');
END;
/
```

DELETE_REALM_FROM_POLICY Procedure

The DELETE_REALM_FROM_POLICY procedure enables you to remove an existing realm from an Oracle Database Vault policy.

You can remove realms from policies any time, regardless of the state (enabled or disabled) of the policy. The change takes effect immediately.

Syntax

```
DBMS_MACADM.DELETE_REALM_FROM_POLICY(
  policy_name    IN VARCHAR2,
  realm_name     IN VARCHAR2);
```

Parameters

Table 23-7 DELETE_REALM_FROM_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view, described in DBA_DV_POLICY View.
realm_name	Realm name. To find existing Database Vault realms in the current database instance, query the DV_REALM view, described in DVSYS.DV$REALM View.

Example

```
BEGIN
 DBMS_MACADM.ADD_DELETE_REALM_FROM_POLICY(
  policy_name    => 'HR_DV_Policy',
  realm_name     => 'HR realm');
```

```
END;
/
```

DROP_POLICY Procedure

The DROP_POLICY procedure enables you to drop an existing Oracle Database Vault policy.

You can remove a policy at any time, regardless of the state (enabled or disabled) of the policy.

Syntax

```
DBMS_MACADM.DROP_POLICY(
  policy_name      IN VARCHAR2);
```

Parameters

Table 23-8 DROP_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view, described in DBA_DV_POLICY View.

Example

```
BEGIN
 DBMS_MACADM.DROP_POLICY(
  policy_name      => 'HR_DV_Policy');
END;
/
```

RENAME_POLICY Procedure

The UPDATE_POLICY_DESCRIPTION procedure enables you to rename an existing Oracle Database Vault policy.

You can rename a policy at any time, regardless of the state (enabled or disabled) of the policy. The change takes effect immediately.

Syntax

```
DBMS_MACADM.RENAME_POLICY(
  policy_name      IN VARCHAR2,
  new_policy_name  IN VARCHAR2);
```

Parameters

Table 23-9 RENAME_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view, described in DBA_DV_POLICY View.
new_policy_name	New policy name, up to 128 characters in mixed case

Example

```
BEGIN
 DBMS_MACADM.RENAME_POLICY(
  policy_name      => 'HR Database Vault Policy',
  new_policy_name  => 'HR_DV_Policy');
END;
/
```

UPDATE_POLICY_DESCRIPTION Procedure

The UPDATE_POLICY_DESCRIPTION procedure enables you to update the description field in an Oracle Database Vault policy.

Syntax

```
DBMS_MACADM.UPDATE_POLICY_DESCRIPTION(
  policy_name  IN VARCHAR2,
  description  IN VARCHAR2 DEFAULT);
```

Parameters

Table 23-10 UPDATE_POLICY_DESCRIPTION Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view, described in DBA_DV_POLICY View.
description	New description of the purpose of the policy, up to 4000 characters in mixed-case

Example

```
BEGIN
 DBMS_MACADM.UPDATE_POLICY_DESCRIPTION(
  policy_name  => 'HR_DV_Policy',
  description  => 'HR schema protection policy');
END;
/
```

UPDATE_POLICY_STATE Procedure

The UPDATE_POLICY_STATE procedure enables you to update the policy_state field in an Oracle Database Vault policy.

Syntax

```
DBMS_MACADM.UPDATE_POLICY_STATE(
  policy_name   IN VARCHAR2,
  policy_state  IN NUMBER,
 pl_sql_stack  IN BOOLEAN DEFAULT);
```

Parameters

Table 23-11 UPDATE_POLICY_STATE Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view, described in DBA_DV_POLICY View.
policy_state	Specifies how the policy is enabled. Possible values are: • DBMS_MACADM.G_ENABLED (1), which enables the policy after you create it • DBMS_MACADM.G_DISABLED (0), which disables the policy after you create it • DBMS_MACADM.G_SIMULATION (2), which sets the policy to simulation mode. In simulation mode, any violations to realms or command rules used in the policy are logged in a designated log table with sufficient information to describe the error, such as the user name or SQL statement used. • DBMS_MACADM.G_PARTIAL (3), which sets the policy to partial mode. In partial mode, the enforcement state of realms or command rules associated with the policy can be changed individually. See About Simulation Mode for more information about simulation mode
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_POLICY_STATE(
  policy_name   => 'HR_DV_Policy',
  policy_state  => DBMS_MACADM.G_DISABLED,
  pl_sql_stack  => TRUE);
END;
/
```

24

Oracle Database Vault API Reference

Oracle Database Vault provides a rich set of APIs, both in PL/SQL packages and in standalone procedures.

- DBMS_MACADM PL/SQL Package Contents
  The DBMS_MACADM package enables you to configure the realms, factors, rule sets, command rules, secure application roles, and Oracle Label Security policies.
- DBMS_MACSEC_ROLES PL/SQL Package Contents
  The DBMS_MACSEC_ROLES package enables you to check and set Oracle Database Vault secure application roles.
- DBMS_MACUTL PL/SQL Package Contents
  The DBMS_MACUTL PL/SQL package defines constants and utility methods that are commonly used by other Oracle Database Vault packages, such as error handling.
- CONFIGURE_DV PL/SQL Procedure
  The CONFIGURE_DV configures the initial two Oracle Database user accounts, which are granted the DV_OWNER and DV_ACCTMGR roles, respectively.
- DVF PL/SQL Interface Contents
  The DVF schema provides a set of factor-related PL/SQL functions.

DBMS_MACADM PL/SQL Package Contents

The DBMS_MACADM package enables you to configure the realms, factors, rule sets, command rules, secure application roles, and Oracle Label Security policies.

The DBMS_MACADM package is available only for users who have been granted the DV_ADMIN or DV_OWNER role.

DBMS_MACADM Realm Procedures

Table 24-1 lists the realm procedures in the DBMS_MACADM package.

Table 24-1 DBMS_MACADM Realm Procedures

Procedure	Description
ADD_AUTH_TO_REALM procedure	Authorizes a user or role to access a realm as an owner or a participant
ADD_OBJECT_TO_REALM procedure	Registers a set of objects for realm protection
CREATE_REALM procedure	Creates a realm
DELETE_AUTH_FROM_REALM procedure	Removes the authorization of a user or role to access a realm
DELETE_OBJECT_FROM_REALM procedure	Removes a set of objects from realm protection

Table 24-1 (Cont.) DBMS_MACADM Realm Procedures

Procedure	Description
DELETE_REALM procedure	Deletes a realm, including its related Database Vault configuration information that specifies who is authorized and what objects are protected
DELETE_REALM_CASCADE procedure	Deletes a realm, including its related Database Vault configuration information that specifies who is authorized and what objects are protected
RENAME_REALM procedure	Renames a realm. The name change takes effect everywhere the realm is used.
UPDATE_REALM procedure	Updates a realm
UPDATE_REALM_AUTH procedure	Updates the authorization of a user or role to access a realm

DBMS_MACADM Rule Set and Rule Procedures

Table 24-2 lists the rule set and rule procedures in the DBMS_MACADM package.

Table 24-2 DBMS_MACADM Rule Set and Rule Procedures

Procedure	Description
CREATE_RULE_SET procedure	Creates a rule set
RENAME_RULE_SET procedure	Renames a rule set. The name change takes effect everywhere the rule set is used.
DELETE_RULE_FROM_RULE_SET procedure	Deletes a rule from a rule set
DELETE_RULE_SET procedure	Deletes a rule set
UPDATE_RULE_SET procedure	Updates a rule set
CREATE_RULE procedure	Creates a rule
ADD_RULE_TO_RULE_SET procedure	Adds a rule to a rule set
DELETE_RULE procedure	Deletes a rule
RENAME_RULE procedure	Renames a rule. The name change takes effect everywhere the rule is used.
UPDATE_RULE procedure	Updates a rule

DBMS_MACADM Command Rule Procedures

Table 24-3 lists the command rule procedures in the DBMS_MACADM package.

Table 24-3 DBMS_MACADM Command Rule Procedures

Procedure	Description
CREATE_COMMAND_RULE procedure	Creates a command rule, associates it with a rule set, and lets you enable the command rule for rule checking with a rule set

Table 24-3 (Cont.) DBMS_MACADM Command Rule Procedures

Procedure	Description
CREATE_CONNECT_COMMAND_RULE procedure	Creates a CONNECT command rule
CREATE_SESSION_EVENT_CMD_RULE procedure	Creates a session event command rule, using the ALTER SESSION SQL statement
CREATE_SYSTEM_EVENT_CMD_RULE procedure	Creates a system event command rule, using the ALTER SYSTEM SQL statement
DELETE_COMMAND_RULE procedure	Drops a command rule declaration
DELETE_CONNECT_COMMAND_RULE procedure	Drops a CONNECT command rule declaration
DELETE_SESSION_EVENT_CMD_RULE procedure	Drops a SESSION_EVENT_CMD command rule declaration
DELETE_SYSTEM_EVENT_CMD_RULE procedure	Drops a SYSTEM_EVENT_CMD command rule declaration
UPDATE_COMMAND_RULE procedure	Updates a command rule declaration
UPDATE_CONNECT_COMMAND_RULE procedure	Updates a CONNECT command rule declaration
UPDATE_SESSION_EVENT_CMD_RULE procedure	Updates a SESSION_EVENT_CMD command rule declaration
UPDATE_SYSTEM_EVENT_CMD_RULE procedure	Updates a SYSTEM_EVENT_CMD command rule declaration

DBMS_MACADM Factor Procedures and Functions

lists the factor procedures and functions in the DBMS_MACADM package.

Table 24-4 DBMS_MACADM Factor Procedures and Functions

Procedure or Function	Description
ADD_FACTOR_LINK procedure	Specifies a parent-child relationship for two factors
ADD_POLICY_FACTOR procedure	Specifies that the label for a factor contributes to the Oracle Label Security label for a policy.
CHANGE_IDENTITY_FACTOR procedure	Associates an identity with a different factor
CHANGE_IDENTITY_VALUE procedure	Updates the value of an identity
CREATE_DOMAIN_IDENTITY procedure	Adds an Oracle Real Application Clusters (Oracle RAC) database node to the domain factor identities and labels it according to the Oracle Label Security policy.
CREATE_FACTOR procedure	Creates a factor
CREATE_FACTOR_TYPE procedure	Creates a factor type
CREATE_IDENTITY procedure	Creates an identity
CREATE_IDENTITY_MAP procedure	Defines a set of tests that are used to derive the identity of a factor from the value of linked child factors (subfactors)

Table 24-4 (Cont.) DBMS_MACADM Factor Procedures and Functions

Procedure or Function	Description
DELETE_FACTOR procedure	Deletes a factor
DELETE_FACTOR_LINK procedure	Removes a parent-child relationship for two factors
DELETE_FACTOR_TYPE procedure	Deletes a factor type
DELETE_IDENTITY procedure	Removes an identity
DELETE_IDENTITY_MAP procedure	Removes an identity map from a factor
DROP_DOMAIN_IDENTITY procedure	Removes an Oracle RAC database node from a domain
GET_INSTANCE_INFO function	Returns information from the SYS.V_$INSTANCE system table about the current database instance; returns a VARCHAR2 value
GET_SESSION_INFO function	Returns information from the SYS.V_$SESSION system table for the current session; returns a VARCHAR2 value
RENAME_FACTOR procedure	Renames a factor. The name change takes effect everywhere the factor is used.
RENAME_FACTOR_TYPE procedure	Renames a factor type. The name change takes effect everywhere the factor type is used.
UPDATE_FACTOR procedure	Updates a factor
UPDATE_FACTOR_TYPE procedure	Updates the description of a factor type
UPDATE_IDENTITY procedure	Updates the trust level of a factor identity

DBMS_MACADM Secure Application Role Procedures

Table 24-5 lists the secure application role procedures in the DBMS_MACADM package.

Table 24-5 DBMS_MACADM Secure Application Role Procedures

Procedure	Description
CREATE_ROLE procedure	Creates an Oracle Database Vault secure application role
DELETE_ROLE procedure	Deletes an Oracle Database Vault secure application role
RENAME_ROLE procedure	Renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.
UNASSIGN_ROLE procedure	Unassigns an Oracle Database Vault secure application role from a user
UPDATE_ROLE procedure	Updates a Oracle Database Vault secure application role

DBMS_MACADM Oracle Label Security Procedures

Table 24-6 lists the Oracle Label Security procedures in the DBMS_MACADM package.

Table 24-6 DBMS_MACADM Oracle Label Security Procedures

Procedure	Description
CREATE_MAC_POLICY procedure	Specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label
CREATE_POLICY_LABEL procedure	Labels an identity within an Oracle Label Security policy
DELETE_MAC_POLICY_CASCADE procedure	Deletes all Oracle Database Vault objects related to an Oracle Label Security policy.
DELETE_POLICY_FACTOR procedure	Removes the factor from contributing to the Oracle Label Security label
DELETE_POLICY_LABEL procedure	Removes the label from an identity within an Oracle Label Security policy
UPDATE_MAC_POLICY procedure	Specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label

DBMS_MACADM Database Vault Policy Procedures

Table 24-7 lists the Database Vault policy procedures in the DBMS_MACADM package.

Table 24-7 DBMS_MACADM Database Vault Policy Procedures

Procedure	Description
ADD_CMD_RULE_TO_POLICY procedure	Adds a command rule to a Database Vault policy
ADD_OWNER_TO_POLICY procedure	Adds an owner to a Database Vault policy
ADD_REALM_TO_POLICY procedure	Adds a realm to a Database Vault policy
CREATE_POLICY procedure	Creates a Database Vault policy
DELETE_CMD_RULE_FROM_POLICY procedure	Deletes a command rule from a Database Vault policy
DELETE_OWNER_FROM_POLICY procedure	Deletes an owner from a Database Vault policy
DELETE_REALM_FROM_POLICY procedure	Deletes a realm from a Database Vault policy
DROP_POLICY procedure	Drops a Database Vault policy
RENAME_POLICY procedure	Renames a Database Vault policy
UPDATE_POLICY_DESCRIPTION procedure	Updates a Database Vault policy description
UPDATE_POLICY_STATE procedure	Updates the enablement status of the a Database Vault policy

DBMS_MACADM General Administrative Procedures

Table 24-8 lists the general administrative procedures in the DBMS_MACADM package.

Table 24-8 DBMS_MACADM General Administrative Procedures

Procedure	Description
ADD_NLS_DATA procedure	Adds a new language to Oracle Database Vault
AUTHORIZE_DATAPUMP_USER procedure	Authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled
AUTHORIZE_DDL procedure	Grants a user authorization to execute data definition language (DDL) statements
AUTHORIZE_MAINTENANCE_USER procedure	Grants a user authorization to perform Information Lifecycle Management (ILM) operations
AUTHORIZE_PROXY_USER procedure	Grants a proxy user authorization to proxy other user accounts
AUTHORIZE_SCHEDULER_USER procedure	Authorizes a user to schedule database jobs when Oracle Database Vault is enabled
AUTHORIZE_TTS_USER procedure	Authorizes a user to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled
UNAUTHORIZE_DATAPUMP_USER procedure	Revokes the authorization that was granted by the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure
UNAUTHORIZE_DDL procedure	Revokes authorization from a user who was granted authorization to execute DDL statements through the DBMS_MACDM.AUTHORIZE_DDL procedure
UNAUTHORIZE_MAINTENANCE_USER procedure	Revokes authorization to perform ILM operations
UNAUTHORIZE_PROXY_USER procedure	Revokes authorization from a user who was granted proxy authorization from the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure
UNAUTHORIZE_SCHEDULER_USER procedure	Revokes authorization that was granted by the DBMS_MACADM.AUTHORIZE_SCHEDULER_USER procedure
UNAUTHORIZE_TTS_USER procedure	Revokes from authorization a user who had been granted authorization to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled
DISABLE_DV procedure	Disables Oracle Database Vault
DISABLE_DV_DICTIONARY_ACCTS procedure	Prevents users from logging into the DVSYS and DFV schema accounts
DISABLE_DV_PATCH_ADMIN	Disables auditing of the DV_PATCH_ADMIN user
DISABLE_ORADEBUG procedure	Disables the use of the ORADEBUG utility in an Oracle Database Vault environment
ENABLE_DV procedure	Enables Oracle Database Vault
ENABLE_DV_DICTIONARY_ACCTS procedure	Enables users to log into the DVSYS and DFV schema accounts
ENABLE_DV_PATCH_ADMIN	Enables auditing of the DV_PATCH_ADMIN user
ENABLE_ORADEBUG procedure	Enables the use of the ORADEBUG utility in an Oracle Database Vault environment

DBMS_MACSEC_ROLES PL/SQL Package Contents

The DBMS_MACSEC_ROLES package enables you to check and set Oracle Database Vault secure application roles.

This package is available to the general database account population.

Table 24-9 lists the contents of the DBMS_MACSEC_ROLES package.

Table 24-9 DBMS_MACSEC_ROLES PL/SQL Package Contents

Procedure or Function	Description
CAN_SET_ROLE function	Checks whether the user invoking the method is authorized to use the specified Oracle Database Vault secure application role. Returns a BOOLEAN value.
SET_ROLE procedure	Issues the SET ROLE statement for an Oracle Database Vault secure application role.

DBMS_MACUTL PL/SQL Package Contents

The DBMS_MACUTL PL/SQL package defines constants and utility methods that are commonly used by other Oracle Database Vault packages, such as error handling.

This package can be run by the general database account population. This allows for security developers to leverage the constants in scripted configuration files. Utility methods such as USER_HAS_ROLE can also be used in Oracle Database Vault rules.

Table 24-10 lists the DBMS_MACUTL package contents.

Table 24-10 DBMS_MACUTL PL/SQL Package Contents

Procedure or Function	Description
CHECK_DVSYS_DML_ALLOWED procedure	Verifies that public-packages are not being bypassed by users updating the Oracle Database Vault configuration
GET_CODE_VALUE function	Looks up the value for a code within a code group.
GET_SECOND function	Returns the seconds in Oracle SS format (00-59). Useful for rule expressions based on time data
GET_MINUTE function	Returns the minute in Oracle MI format (00–59). Useful for rule expressions based on time data
GET_HOUR function	Returns the month in Oracle HH24 format (00–23). Useful for rule expressions based on time data
GET_DAY function	Returns the day in Oracle DD format (01–31). Useful for rule expressions based on time data
GET_MONTH function	Returns the month in Oracle MM format (01–12). Useful for rule expressions based on time data
GET_YEAR function	Returns the year in Oracle YYYY format (0001–9999). Useful for rule expressions based on time data
IS_ALPHA function	Checks whether the character is alphabetic

Table 24-10 (Cont.) DBMS_MACUTL PL/SQL Package Contents

Procedure or Function	Description
IS_DIGIT function	Checks whether the character is numeric
IS_DVSYS_OWNER function	Determines whether a user is authorized to manage the Oracle Database Vault configuration
IS_OLS_INSTALLED function	Returns an indicator regarding whether Oracle Label Security is installed
IS_OLS_INSTALLED_VARCHAR function	Returns an indicator regarding whether Oracle Label Security is installed
USER_HAS_ROLE function	Checks whether a user has a role privilege, directly or indirectly (through another role)
USER_HAS_ROLE_VARCHAR function	Checks whether a user has a role privilege, directly or indirectly (through another role)
USER_HAS_SYSTEM_PRIVILEGE function	Checks whether a user has a system privilege, directly or indirectly (through a role)

CONFIGURE_DV PL/SQL Procedure

The CONFIGURE_DV configures the initial two Oracle Database user accounts, which are granted the DV_OWNER and DV_ACCTMGR roles, respectively.

This procedure is used as part of the registration process for Oracle Database Vault with an Oracle database. You only need to use it once for the database instance.

DVF PL/SQL Interface Contents

The DVF schema provides a set of factor-related PL/SQL functions.

The functions are then available to the general database account population through PL/SQL functions and standard SQL.

Table 24-11 lists the DVF factor functions.

Table 24-11 DVF PL/SQL Interface Contents

Function	Description
F$CLIENT_IP	Returns the IP address of the computer from which the client is connected
F$DATABASE_DOMAIN	Returns the domain of the database as specified in the DB_DOMAIN initialization parameter
F$DATABASE_HOSTNAME	Returns the host name of the computer on which the database instance is running
F$DATABASE_INSTANCE	Returns the database instance identification number of the current database instance
F$DATABASE_IP	Returns the IP address of the computer on which the database instance is running

Table 24-11 (Cont.) DVF PL/SQL Interface Contents

Function	Description
F$DATABASE_NAME	Returns the name of the database as specified in the DB_NAME initialization parameter
F$DOMAIN	Returns a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level
F$ENTERPRISE_IDENTITY	Returns the enterprise-wide identity for a user
F$IDENTIFICATION_TYPE	Returns the way the schema of a user was created in the database. Specifically, it reflects the IDENTIFIED clause in the CREATE USER or ALTER USER syntax.
F$LANG	Returns the ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter
F$LANGUAGE	Returns the language and territory currently used by your session, in VARCHAR2 data type, along with the database character set
F$MACHINE	Returns the computer (host) name for the database client that established the database session.
F$NETWORK_PROTOCOL	Returns the network protocol being used for communication, as specified in the PROTOCOL=*protocol* portion of the connect string
F$PROXY_ENTERPRISE_IDENTITY	Returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user
F$SESSION_USER	Returns the database user name by which the current user is authenticated

25

Oracle Database Vault Data Dictionary Views

You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

- DBA_DV_FACTOR_LINK View
  The DBA_DV_FACTOR_LINK data dictionary view shows the relationships of each factor whose identity is determined by the association of child factors.
- DBA_DV_IDENTITY View
  The DBA_DV_IDENTITY data dictionary view lists the identities for each factor.
- DBA_DV_IDENTITY_MAP View
  The DBA_DV_IDENTITY_MAP data dictionary view lists the mappings for each factor identity.
- DBA_DV_JOB_AUTH View
  The DBA_DV_JOB_AUTH data dictionary view lists the authorizations for using Oracle Scheduler in an Oracle Database Vault environment.
- DBA_DV_MAC_POLICY View
  The DBA_DV_MAC_POLICY data dictionary view lists the Oracle Label Security policies defined for use with Oracle Database Vault.
- DBA_DV_MAC_POLICY_FACTOR View
  The DBA_DV_MAC_POLICY data dictionary view lists the factors that are associated with Oracle Label Security policies.
- DBA_DV_MAINTENANCE_AUTH View
  The DBA_DV_MAINTENANCE_AUTH data dictionary view provides information about the configuration of Oracle Database Vault authorizations to use Information Life Management (ILM) features.
- DBA_DV_ORADEBUG View
  The DBA_DV_ORADEBUG data dictionary view indicates whether users can use the ORADEBUG utility in an Oracle Database Vault environment.
- DBA_DV_PATCH_ADMIN_AUDIT View
  The DBA_DV_PATCH_ADMIN_AUDIT data dictionary view indicates if auditing has been enabled or disabled for the user who has been granted the DV_ADMIN_PATCH role.
- DBA_DV_POLICY View
  The DBA_DV_POLICY data dictionary view lists the Oracle Database Vault policies that were created in the current database instance.
- DBA_DV_POLICY_LABEL View
  The DBA_DV_POLICY_LABEL data dictionary view lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.
- DBA_DV_POLICY_OBJECT View
  The DBA_DV_POLICY_OBJECT data dictionary view lists information about the objects that are protected by Oracle Database Vault policies in the current database instance.
- DBA_DV_POLICY_OWNER View
  The DBA_DV_POLICY_OWNER data dictionary view lists the owners of Oracle Database Vault policies that were created in the current database instance.
- DBA_DV_PREPROCESSOR_AUTH View
  The DBA_DV_PREPROCESSOR_AUTH data dictionary view shows users who have been granted authorization to execute preprocessor programs through external tables.
- DBA_DV_PROXY_AUTH View
  The DBA_DV_PROXY_AUTH data dictionary view lists the proxy users and schemas that were specified by the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure.

- DBA_DV_PUB_PRIVS View
  The DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in Oracle Database Vault Administrator.
- DBA_DV_REALM View
  The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.
- DBA_DV_REALM_AUTH View
  The DBA_DV_REALM_AUTH data dictionary view lists database user account or role authorization (GRANTEE) who can access realm objects.
- DBA_DV_REALM_OBJECT View
  The DBA_DV_REALM_OBJECT data dictionary view lists the database schemas, or subsets of schemas, that are secured by the realms.
- DBA_DV_ROLE View
  The DBA_DV_ROLE data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.
- DBA_DV_RULE View
  The DBA_DV_RULE data dictionary view lists the rules that have been defined.
- DBA_DV_RULE_SET View
  The DBA_DV_RULE_SET data dictionary view lists the rules sets that have been created.
- DBA_DV_RULE_SET_RULE View
  The DBA_DV_RULE_SET_RULE data dictionary view lists rules that are associated with existing rule sets.
- DBA_DV_STATUS View
  The DBA_DV_STATUS data dictionary view shows the status of Oracle Database Vault being enabled and configured.
- DBA_DV_SIMULATION_LOG View
  The DBA_DV_SIMULATION_LOG data dictionary view captures simulation log information for realms and command rules that have had simulation mode enabled.
- DBA_DV_TTS_AUTH View
  The DBA_DV_TTS_AUTH data dictionary view lists users who have been granted authorization through the DBMS_MACADM.AUTHORIZE_TTS_USER procedure to perform Oracle Data Pump transportable operations.
- DBA_DV_USER_PRIVS View
  The DBA_DV_USER_PRIVS data dictionary view lists the privileges for a database user account excluding privileges granted through the PUBLIC role.
- DBA_DV_USER_PRIVS_ALL View
  The DBA_DV_USER_PRIVS_ALL data dictionary view lists the privileges for a database account including privileges granted through PUBLIC.
- DVSYS.DV$CONFIGURATION_AUDIT View
  The DVSYS.DV$CONFIGURATION_AUDIT data dictionary view captures DVSYS.AUDIT_TRAIL$ table audit trail records.
- DVSYS.DV$ENFORCEMENT_AUDIT View
  The DVSYS.DV$ENFORCEMENT_AUDIT data dictionary view provides information about enforcement-related audits from the DVSYS.AUDIT_TRAIL$ table.

- DVSYS.DV$REALM View
  The DVSYS.DV$REALM data dictionary view describes settings that were used to create Oracle Database Vault realms, such as which audit options have been assigned or whether the realm is a mandatory realm.
- DVSYS.POLICY_OWNER_COMMAND_RULE View
  The DVSYS.POLICY_OWNER_COMMAND_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the command rules that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_POLICY View
  The DVSYS.POLICY_OWNER_POLICY data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners.
- DVSYS.POLICY_OWNER_REALM View
  The POLICY_OWNER_REALM data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the realms that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_REALM_AUTH View
  The DVSYS.POLICY_OWNER_REALM_AUTH data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the authorization that was granted to realms that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_REALM_OBJECT View
  The DVSYS.POLICY_OWNER_REALM_OBJECT data dictionary view enables users to find information about the objects that have been added to realms that are associated with Database Vault policies, such as. Only users who have been granted the DV_POLICY_OWNER role can query this view.
- DVSYS.POLICY_OWNER_RULE View
  The DVSYS.POLICY_OWNER_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rules that have been associated with rule sets in Database Vault policies, such as the rule name and its expression. Only users who have been granted the DV_POLICY_OWNER role can query this view.
- DVSYS.POLICY_OWNER_RULE_SET View
  The DVSYS.POLICY_OWNER_RULE_SET data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_RULE_SET_RULE View
  The DVSYS.POLICY_OWNER_RULE_SET_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that contain rules used in Database Vault policies.
- AUDSYS.DV$CONFIGURATION_AUDIT View
  The AUDSYS.DV$CONFIGURATION_AUDIT view is almost the same as the DVSYS.DV$CONFIGURATION_AUDIT view except that it captures unified audit trail Database Vault audit records.
- AUDSYS.DV$ENFORCEMENT_AUDIT View
  The AUDSYS.DV$ENFORCEMENT_AUDIT view is almost the same as the DVSYS.DV$ENFORCEMENT_AUDIT view except that it captures unified audit trail Database Vault audit records.

About the Oracle Database Vault Data Dictionary Views

Oracle Database Vault provides a set of DBA-style data dictionary views that can be accessed through the DV_SECANALYST role or the DV_ADMIN role.

These views provide access to the various underlying Oracle Database Vault tables in the DVSYS and LBACSYS schemas without exposing the primary and foreign key columns that may be present. These views are intended for the database administrative user to report on the state of the Oracle Database Vault configuration without having to perform the joins required to get the labels for codes that are stored in the core tables or from the related tables.

> **See Also:**
>
> Oracle Database Vault Reports if you are interested in running reports on Oracle Database Vault

CDB_DV_STATUS View

The CDB_DV_STATUS data dictionary view shows the status of Oracle Database Vault being enabled and configured in a multitenant environment.

Only Oracle Database administrative users, such users who have been granted the DBA role, can query this view. Database Vault administrators do not have access to this view.

For example:

```
SELECT * FROM CDB_DV_STATUS;
```

Output similar to the following appears:

```
NAME                 STATUS CON_ID
-------------------- ------ ------
DV_CONFIGURE_STATUS  TRUE        0
DV_ENABLE_STATUS     TRUE        0
```

Related Views

- DBA_DV_STATUS View

Column	Datatype	Null	Description
NAME	VARCHAR2(19)	NOT NULL	Shows either of the following settings: • DV_CONFIGURE_STATUS shows whether Oracle Database Vault has been configured, that is, with the CONFIGURE_DV procedure. • DV_ENABLE_STATUS shows whether Oracle Database Vault has been enabled, that is, with the DBMS_MACADM.ENABLE_DV procedure.

Column	Datatype	Null	Description
STATUS	VARCHAR2(64)	NOT NULL	TRUE means that Oracle Database Vault is configured or enabled; FALSE means that it is not.
CON_ID	NUMBER	NOT NULL	The identification number of the container in which Oracle Database Vault is used

DBA_DV_CODE View

The DBA_DV_CODE data dictionary view lists generic lookup codes for the user interface, error messages, and constraint checking.

These codes are used for the user interface, views, and for validating input in a translatable fashion.

For example:

```
SELECT CODE, VALUE FROM DBA_DV_CODE WHERE CODE_GROUP = 'BOOLEAN';
```

Output similar to the following appears:

```
CODE    VALUE
------- --------
Y       True
N       False
```

Column	Datatype	Null	Description
CODE_GROUP	VARCHAR(128)	NOT NULL	Displays one of the code groups that are listed in Table 25-1
CODE	VARCHAR(128)	NOT NULL	Boolean code used; either Y (Yes) or N (No).
VALUE	VARCHAR(4000)	NULL	Boolean value used; either True if the Boolean code is Y or False if the Boolean code is N.
LANGUAGE	VARCHAR(3)	NOT NULL	Language for this installation of Oracle Database Vault. Supported languages are as follows: • en: English • de: German • es: Spanish • fr: French • it: Italian • ja: Japanese • ko: Korean • pt_BR: Brazilian Portuguese • zh_CN: Simplified Chinese • zh_TW: Traditional Chinese
DESCRIPTION	VARCHAR(1024)	NULL	Brief description of the code group.

Table 25-1 describes the possible values from the CODE_GROUP column in the DBA_DV_CODE data dictionary view.

Table 25-1 DBA_DV_CODE View CODE_GROUP Values

CODE_GROUP Name	Description
AUDIT_EVENTS	Contains the action numbers and action names that are used for the custom event audit trail records
BOOLEAN	A simple Yes or No or True or False lookup
DB_OBJECT_TYPE	The database object types that can be used for realm objects and command authorizations
SQL_CMDS	The DDL commands that can be protected through command rules
FACTOR_AUDIT	The auditing options for factor retrieval processing
FACTOR_EVALUATE	The evaluation options (by session or by access) for factor retrieval
FACTOR_FAIL	The options for propagating errors when a factor retrieval method fails
FACTOR_IDENTIFY	The options for determining how a factor identifier is resolved (for example, by method or by factors)
FACTOR_LABEL	The options for determining how a factor identifier is labeled in the session establishment phase
LABEL_ALG	The algorithms that can be used to determine the maximum session label for a database session for each policy. See Table 20-2 for a listing of the Oracle Label Security merge algorithm codes.
OPERATORS	The Boolean operators that can be used for identity maps
REALM_AUDIT	The options for auditing realm access or realm violations
REALM_OPTION	The options for ownership of a realm
RULESET_AUDIT	The options for auditing rule set execution or rule set errors
RULESET_EVALUATE	The options for determining the success or failure of a rule set based on all associated rules being true or any associated rule being true
RULESET_EVENT	The options to invoke a custom event handler when a rule set evaluates to Succeeds or Fails
RULESET_FAIL	The options to determine the run-time visibility of a rule set failing

DBA_DV_COMMAND_RULE View

The DBA_DV_COMMAND_RULE data dictionary view lists the SQL statements that are protected by command rules.

See Configuring Command Rules , for more information about command rules.

For example:

```
SELECT COMMAND, RULE_SET_NAME FROM DBA_DV_COMMAND_RULE;
```

Output similar to the following appears:

```
COMMAND         RULE_SET_NAME
--------------- ----------------------------
GRANT           Can Grant VPD Administration
REVOKE          Can Grant VPD Administration
ALTER SYSTEM    Allow System Parameters
ALTER USER      Can Maintain Own Account
CREATE USER     Can Maintain Account/Profiles
DROP USER       Can Maintain Account/Profiles
CREATE PROFILE  Can Maintain Account/Profiles
DROP PROFILE    Can Maintain Account/Profiles
ALTER PROFILE   Can Maintain Account/Profiles
```

Column	Datatype	Null	Description
COMMAND	VARCHAR(128)	NOT NULL	Name of the command rule. For a list of default command rules, see Default Command Rules.
CLAUSE_NAME	VARCHAR(100)	NOT NULL	A clause from either the ALTER SYSTEM or ALTER SESSION SQL statement, which was used to create the command rule. For example, you it could list the SET clause for the ALTER SESSION statement. For a full list of possible clause values, see the following topics: • Table 17-2 • Table 17-3
PARAMETER_NAME	VARCHAR(128)	NOT NULL	A parameter from the ALTER SYSTEM or ALTER SESSION command rule CLAUSE_NAME setting
EVENT_NAME	VARCHAR(128)	NOT NULL	An event that the ALTER SYSTEM or ALTER SESSION command rule defines
COMPONENT_NAME	VARCHAR(128)	NOT NULL	A component of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule.
ACTION_NAME	VARCHAR(128)	NOT NULL	An action of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set associated with this command rule.
OBJECT_OWNER	VARCHAR(128)	NOT NULL	The owner of the object that the command rule affects.
OBJECT_NAME	VARCHAR(128)	NOT NULL	The name of the database object the command rule affects (for example, a database table).
ENABLED	VARCHAR(1)	NOT NULL	Possible values are as follows: • Y indicates the command rule is enabled • N indicates it is disabled • S indicates it is in simulation mode
PRIVILEGE_SCOPE	NUMBER	NOT NULL	Obsolete column
COMMON	VARCHAR(3)	NOT NULL	For a multitenant environment, indicates whether the command rule is local or common. Possible values are: • YES if the command rule is common • NO if the command rule is local

Column	Datatype	Null	Description
INHERITED	VARCHAR(3)	NOT NULL	Shows the inheritance status of the command rule, when the COMMON column output is YES. Values are as follows: • YES means that the command rule was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the command rule is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ID#	NUMBER	NOT NULL	The ID number of the command rule, which is automatically generated when the command rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the command rule is a default (that is, Oracle-supplied) command rule or a user-created command rule. Possible values are: • YES if the command rule is a default command rule • NO if the command rule is a user-created command rule
PL_SQL_STACK	VARCHAR(3)	NULL	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

DBA_DV_DATAPUMP_AUTH View

The DBA_DV_DATAPUMP_AUTH data dictionary view lists the authorizations for using Oracle Data Pump in an Oracle Database Vault environment.

See Using Oracle Data Pump with Oracle Database Vault for more information.

For example:

```
SELECT * FROM DBA_DV_DATAPUMP_AUTH WHERE GRANTEE = 'PRESTON';
```

Output similar to the following appears:

```
GRANTEE SCHEMA OBJECT
------- ------ -------
PRESTON OE     ORDERS
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Data Pump authorization
SCHEMA	VARCHAR2(128)	NOT NULL	Name of the schema on which the user GRANTEE is authorized to perform Data Pump operations

Column	Datatype	Null	Description
OBJECT	VARCHAR2(128)	NOT NULL	Name of the object within the schema specified by the SCHEMA parameter on which the GRANTEE user has Data Pump authorization (such as a table)

DBA_DV_DBCAPTURE_AUTH View

The DBA_DV_DBCAPTURE_AUTH data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload capture operations.

See Using Oracle Database Replay with Oracle Database Vault for more information.

For example:

```
SELECT * FROM DBA_DV_DBCAPTURE_AUTH WHERE GRANTEE = 'PFITCH';
```

Output similar to the following appears:

```
GRANTEE
-------
PFITCH
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Database Replay workload capture authorization

DBA_DV_DBREPLAY View

The DBA_DV_DBREPLAY_AUTH data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload replay operations.

See Using Oracle Database Replay with Oracle Database Vault for more information.

For example:

```
SELECT * FROM DBA_DV_DBREPLAY_AUTH WHERE GRANTEE = 'PFITCH';
```

Output similar to the following appears:

```
GRANTEE
-------
PFITCH
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Database Replay workload replay authorization

DBA_DV_DDL_AUTH View

The DBA_DV_DDL data dictionary view lists the users and schemas that were specified by the DBMS_MACADM.AUTHORIZE_DDL procedure.

This procedure grants a user authorization to execute Data Definition Language (DDL) statements.

For example:

```
SELECT * FROM DBA_DV_DDL_AUTH WHERE GRANTEE = 'psmith';
```

Output similar to the following appears:

```
GRANTEE SCHEMA
------- ------
PSMITH  HR
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted DDL authorization
SCHEMA	VARCHAR2(128)	NOT NULL	Name of the schema on which the user GRANTEE is authorized to perform DDL operations

See Also:

- AUTHORIZE_DDL Procedure
- UNAUTHORIZE_DDL Procedure

DBA_DV_DICTIONARY_ACCTS View

The DBA_DV_DICTIONARY_ACCTS data dictionary view indicates whether users can directly log into the DVSYS and DVF schema accounts.

For example:

```
SELECT * FROM DBA_DV_DICTIONARY_ACCTS;
```

Output similar to the following appears:

```
STATE
-------
ENABLED
```

Column	Datatype	Null	Description
STATE	VARCHAR2(8)	NOT NULL	Describes whether users can log directly into the DVSYS and DVF schemas. Possible values are: • ENABLED means that users can log directly into the DVSYS and DVF schemas • DISABLED means that users cannot log directly into the DVSYS and DVF schemas

DBA_DV_FACTOR View

The DBA_DV_FACTOR data dictionary view lists the existing factors in the current database instance.

For example:

```
SELECT NAME, GET_EXPR FROM DBA_DV_FACTOR WHERE NAME = 'Session_User';
```

Output similar to the following appears:

```
NAME          GET_EXPR
------------- ------------------------------------------
Session_User  UPPER(SYS_CONTEXT('USERENV', 'SESSION_USER'))
```

Related Views

- DBA_DV_FACTOR_LINK View
- View

Column	Datatype	Null	Description
NAME	VARCHAR2(128)	NOT NULL	Name of the factor. See Default Factors for a list of default factors.
DESCRIPTION	VARCHAR2(4000)	NULL	Description of the factor.
FACTOR_TYPE_NAME	VARCHAR2(128)	NOT NULL	Category of the factor, which is used to classify the purpose of the factor.
ASSIGN_RULE_SET_NAME	VARCHAR2(128)	NULL	Rule set used to control the identify of the factor.
GET_EXPR	VARCHAR2(1024)	NULL	PL/SQL expression that retrieves the identity of a factor.
VALIDATE_EXPR	VARCHAR2(1024)	NULL	PL/SQL expression used to validate the identify of the factor. It returns a Boolean value.
IDENTIFIED_BY	NUMBER	NOT NULL	Determines the identity of a factor, based on the expression listed in the GET_EXPR column. Possible values are: • 0: By constant • 1: By method • 2: By factors

Column	Datatype	Null	Description
IDENTIFIED_BY_MEANING	VARCHAR2(4000)	NULL	Provides a text description for the corresponding value in the IDENTIFIED_BY column. Possible values are: • By Constant: If IDENTIFIED_COLUMN is 0 • By Method: If IDENTIFIED_COLUMN is 1 • By Factors: If IDENTIFIED_COLUMN is 2
LABELED_BY	NUMBER	NOT NULL	Determines the labeling the factor: • 0: Labels the identities for the factor directly from the labels associated with an Oracle Label Security policy • 1: Derives the factor identity label from the labels of its child factor identities.
LABELED_BY_MEANING	VARCHAR2(4000)	NULL	Provides a text description for the corresponding value in the LABELED_BY column. Possible values are: • By Self: If LABELED_BY column is 0 • By Factors: If LABELED_BY column is 1
EVAL_OPTIONS	NUMBER	NOT NULL	Determines how the factor is evaluated when the user logs on: • 0: When the database session is created • 1: Each time the factor is accessed • 2: On start-up
EVAL_OPTIONS_MEANING	VARCHAR2(4000)	NULL	Provides a text description for the corresponding value in the EVAL_OPTIONS column. Possible values are: • For Session: If EVAL_OPTIONS is 0 • By Access: If EVAL_OPTIONS is 1 • On Startup: If EVAL_OPTIONS is 2
AUDIT_OPTIONS	NUMBER	NOT NULL	Option for auditing the factor if you want to generate a custom Oracle Database Vault audit record. Possible values are: • 0: No auditing set • 1: Always audits • 2: Audits if get_expr returns an error • 4: Audits if get_expr is null • 8: Audits if the validation procedure returns an error • 16: Audits if the validation procedure is false • 32: Audits if there is no trust level set • 64: Audits if the trust level is negative.
FAIL_OPTIONS	NUMBER	NOT NULL	Options for reporting factor errors: • 1: Shows an error message. • 2: Does not show an error message.

Column	Datatype	Null	Description
FAIL_OPTIONS_MEANING	VARCHAR2(4000)	NULL	Provides a text description for the corresponding value in the FAIL_OPTIONS column. Possible values are: • Show Error Message • Do Not Show Error Message:
ID#	NUMBER	NOT NULL	The ID number of the factor, which is automatically generated when the factor is created
ORACLE_SUPPLIED	VARCHAR(3)	NOT NULL	Indicates whether the factor is a default (that is, Oracle-supplied) factor or a user-created factor. Possible values are: • YES if the factor is a default factor • NO if the factor is a user-created factor

DBA_DV_FACTOR_TYPE View

The DBA_DV_FACTOR_TYPE data dictionary view lists the names and descriptions of factor types used in the system.

For example:

```
SELECT * FROM DBA_DV_FACTOR_TYPE WHERE NAME = 'Hostname';
```

Output similar to the following appears:

```
NAME      DESCRIPTION
--------- --------------------------------------------------------------------
Time      Time-based factor
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_FACTOR_LINK View

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Name of the factor type.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the factor type.

DBA_DV_FACTOR_LINK View

The DBA_DV_FACTOR_LINK data dictionary view shows the relationships of each factor whose identity is determined by the association of child factors.

This view contains one entry for each parent factor and child factor. You can use this view to resolve the relationships from the factor links to identity maps.

For example:

```
SELECT PARENT_FACTOR_NAME, CHILD_FACTOR_NAME FROM DBA_DV_FACTOR_LINK;
```

Output similar to the following appears:

```
PARENT_FACTOR_NAME             CHILD_FACTOR_NAME
------------------------------ ------------------------------
Domain                         Database_Instance
Domain                         Database_IP
Domain                         Database_Hostname
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_FACTOR_TYPE View

Column	Datatype	Null	Description
PARENT_FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the parent factor
CHILD_FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the child factor of the parent factor
LABEL_IND	VARCHAR(1)	NOT NULL	Indicates whether the child factor that is linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Possible values are: • Y (for Yes) • N (for No)

DBA_DV_IDENTITY View

The DBA_DV_IDENTITY data dictionary view lists the identities for each factor.

For example:

```
SELECT * FROM DBA_DV_IDENTITY WHERE VALUE = 'GLOBAL SHARED';
```

Output similar to the following appears, assuming you have created only one factor identity:

```
FACTOR_NAME          VALUE          TRUST_LEVEL
---------------      -------------- ------------
Identification_Type  GLOBAL SHARED  1
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_IDENTITY_MAP View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the factor.
VALUE	VARCHAR(1024)	NOT NULL	Value of the factor.
TRUST_LEVEL	NUMBER	NOT NULL	Number that indicates the magnitude of trust relative to other identities for the same factor.

DBA_DV_IDENTITY_MAP View

The DBA_DV_IDENTITY_MAP data dictionary view lists the mappings for each factor identity.

The view includes mapping factors that are identified by other factors to combinations of parent-child factor links. For each factor, the maps are joined by the OR operation, and for different factors, the maps are joined by the AND operation.

You can use this view to resolve the identity for factors that are identified by other factors (for example, a domain) or for factors that have continuous domains (for example, Age or Temperature).

For example:

```
SELECT FACTOR_NAME, IDENTITY_VALUE FROM DBA_DV_IDENTITY_MAP;
```

Output similar to the following appears:

```
FACTOR_NAME      IDENTITY_VALUE
---------------- --------------------
Sector2_Program  Accounting-Sensitive
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_IDENTITY View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(128)	NOT NULL	Factor the identity map is for.
IDENTITY_VALUE	VARCHAR(1024)	NOT NULL	Value the factor assumes if the identity map evaluates to TRUE.
OPERATION_CODE	VARCHAR(128)	NOT NULL	Descriptive name of the operation in the OPERATION_VALUE column.
OPERATION_VALUE	VARCHAR(4000)	NULL	Relational operator for the identity map (for example, <, >, =, and so on).
OPERAND1	VARCHAR(1024)	NULL	Left operand for the relational operator; refers to the low value you enter.
OPERAND2	VARCHAR(1024)	NULL	Right operand for the relational operator; refers to the high value you enter.
PARENT_FACTOR_NAME	VARCHAR(128)	NULL	The parent factor link to which the map is related.
CHILD_FACTOR_NAME	VARCHAR(128)	NULL	The child factor link to which the map is related.
LABEL_IND	VARCHAR(1)	NULL	Indicates whether the child factor being linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Possible values are: • Y (for Yes) • N (for No)

DBA_DV_JOB_AUTH View

The DBA_DV_JOB_AUTH data dictionary view lists the authorizations for using Oracle Scheduler in an Oracle Database Vault environment.

For example:

```
SELECT * FROM DBA_DV_JOB_AUTH WHERE GRANTEE = 'PRESTON';
```

Output similar to the following appears:

```
GRANTEE SCHEMA
------- ------
PRESTON OE
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Oracle Scheduler authorization
SCHEMA	VARCHAR2(128)	NOT NULL	Name of the schema on which the user GRANTEE is authorized to perform Oracle Scheduler operations

DBA_DV_MAC_POLICY View

The DBA_DV_MAC_POLICY data dictionary view lists the Oracle Label Security policies defined for use with Oracle Database Vault.

For example:

```
SELECT POLICY_NAME, ALGORITHM_CODE, ALGORITHM_MEANING
 FROM DBA_DV_MAC_POLICY;
```

Output similar to the following appears:

```
POLICY_NAME     ALGORITHM_CODE    ALGORITHM_MEANING
--------------- ----------------- ------------------------------
ACCESS_DATA     LUI               Minimum Level/Union/Intersection
```

Related Views

- DBA_DV_MAC_POLICY_FACTOR View
- DBA_DV_POLICY_LABEL View

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(128)	NOT NULL	Name of the policy.
ALGORITHM_CODE	VARCHAR(128)	NOT NULL	Merge algorithm code used for the policy. See Table 20-2 for a listing of algorithm codes.
ALGORITHM_MEANING	VARCHAR(4000)	NULL	Provides a text description for the corresponding value in the ALGORITHM_CODE column. See Table 20-2 for a listing of algorithm code descriptions.
ERROR_LABEL	VARCHAR(4000)	NULL	Label specified for initialization errors, to be set when a configuration error or run-time error occurs during session initialization.

DBA_DV_MAC_POLICY_FACTOR View

The DBA_DV_MAC_POLICY data dictionary view lists the factors that are associated with Oracle Label Security policies.

You can use this view to determine what factors contribute to the maximum session label for each policy using the DBA_DV_MAC_POLICY view.

For example:

```
SELECT * FROM DBA_DV_MAC_POLICY_FACTOR;
```

Output similar to the following appears:

```
FACTOR_NAME    MAC_POLICY_NAME
-------------- -----------------
App_Host_Name  Access Locations
```

Related Views

- DBA_DV_MAC_POLICY View
- DBA_DV_POLICY_LABEL View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the factor
MAC_POLICY_NAME	VARCHAR(128)	NOT NULL	Name of the Oracle Label Security policy associated with this facto

DBA_DV_MAINTENANCE_AUTH View

The DBA_DV_MAINTENANCE_AUTH data dictionary view provides information about the configuration of Oracle Database Vault authorizations to use Information Life Management (ILM) features.

For example:

```
SELECT GRANTEE, ACTION STATE FROM DBA_DV_MAINTENANCE_AUTH;
```

Output similar to the following appears:

```
GRANTEE                  ACTION
------------------------ --------
PSMITH                   ILM
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR(128)	NOT NULL	Name of the grantee
SCHEMA	VARCHAR(128)	NOT NULL	Schema name or % (for all schemas)
OBJECT	VARCHAR(128)	NOT NULL	Object name or % (for all objects in a schema)
OBJECT_TYPE	VARCHAR(30)	NOT NULL	Object type
ACTION	VARCHAR(30)	NOT NULL	Maintenance action ILM for ILM operations

DBA_DV_ORADEBUG View

The DBA_DV_ORADEBUG data dictionary view indicates whether users can use the ORADEBUG utility in an Oracle Database Vault environment.

For example:

```
SELECT * FROM DBA_DV_ORADEBUG;
```

Output similar to the following appears:

```
STATE
--------
DISABLED
```

Column	Datatype	Null	Description
STATE	VARCHAR2(8)	NOT NULL	Describes whether the ORADEBUG utility can be used in a Database Vault-enabled environment. Possible values are: • ENABLED means that users can run the ORADEBUG utility • DISABLED means that users cannot run the ORADEBUG utility

DBA_DV_PATCH_ADMIN_AUDIT View

The DBA_DV_PATCH_ADMIN_AUDIT data dictionary view indicates if auditing has been enabled or disabled for the user who has been granted the DV_ADMIN_PATCH role.

The DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT procedure enables this type of auditing.

For example:

```
SELECT * FROM DBA_DV_PATCH_ADMIN_AUDIT;
```

Output similar to the following appears:

```
STATE
--------
DISABLED
```

Column	Datatype	Null	Description
STATE	VARCHAR2(8)	NOT NULL	Describes whether auditing has been enabled or disabled for the DV_ADMIN_PATCH role user. Possible values are: • ENABLED means that the auditing has been enabled • DISABLED means that the auditing has been disabled

See Also:

- ENABLE_DV_PATCH_ADMIN_AUDIT Procedure
- DISABLE_DV_PATCH_ADMIN_AUDIT Procedure

DBA_DV_POLICY View

The DBA_DV_POLICY data dictionary view lists the Oracle Database Vault policies that were created in the current database instance.

For example:

```
SELECT POLICY_NAME, STATE FROM DBA_DV_POLICY
  WHERE STATE = 'ENABLED';
```

Output similar to the following appears:

```
POLICY_NAME                      STATE
-------------------------------- -------
Oracle Account Management Controls ENABLED
Oracle System Protection Controls  ENABLED
```

Related Views

- DBA_DV_POLICY_OWNER View
- DBA_DV_POLICY_OBJECT View
- DBA_DV_SIMULATION_LOG View
- DVSYS.POLICY_OWNER_POLICY View

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(128)	NOT NULL	Names of the Oracle Database Vault policies that have been created. See Default Oracle Database Vault Policiesfor a listing of default policies.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the policy that was created
STATE	VARCHAR(8)	NULL	Specifies whether the policy is enabled. Possible values are: • ENABLED • DISABLED • SIMULATION
ID#	VARCHAR(1)	NOT NULL	Is a system-generated ID that was assigned to the policy when the policy was created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the policy is a default Oracle Database Vault policy
PL_SQL_STACK	VARCHAR(3)	NULL	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

DBA_DV_POLICY_LABEL View

The DBA_DV_POLICY_LABEL data dictionary view lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.

For example:

```
SELECT * FROM DBA_DV_POLICY_LABEL;
```

Output similar to the following appears:

```
IDENTITY_VALUE    FACTOR_NAME     POLICY_NAME       LABEL
----------------- -------------  ---------------  ---------
App_Host_Name     Sect2_Fin_Apps  Access Locations  Sensitive
```

Related Views

- DBA_DV_MAC_POLICY View
- DBA_DV_MAC_POLICY_FACTOR View

Column	Datatype	Null	Description
IDENTITY_VALUE	VARCHAR(1024)	NOT NULL	Name of the factor identifier.
FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the factor associated with the factor identifier.
POLICY_NAME	VARCHAR(128)	NOT NULL	Name of the Oracle Label Security policy associated with this factor.
LABEL	VARCHAR(4000)	NOT NULL	Name of the Oracle Label Security label associated with the policy.

DBA_DV_POLICY_OBJECT View

The DBA_DV_POLICY_OBJECT data dictionary view lists information about the objects that are protected by Oracle Database Vault policies in the current database instance.

For example:

```
SELECT POLICY_NAME, OBJECT_TYPE FROM DBA_DV_POLICY_OBJECT WHERE POLICY_NAME LIKE
'%Protection Controls';
```

Output similar to the following appears:

```
POLICY_NAME                       OBJECT_TYPE
--------------------------------- ------------
Oracle System Protection Controls  REALM
```

Related Views

- DBA_DV_POLICY View
- DBA_DV_POLICY_OWNER View

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(128)	NOT NULL	Names of the Oracle Database Vault policies that have been created. See Default Oracle Database Vault Policies for a listing of default policies.
OBJECT_TYPE	VARCHAR(12)	NULL	Type of object that is being protected, such as REALM
COMMAND	VARCHAR(128)	NULL	Name of the command rules that are protected by Database Vault policies
COMMAND_OBJ_OWNER	VARCHAR(128)	NULL	Names of object owners that are associated with Database Vault policies
COMMAND_OBJ_NAME	VARCHAR(128)	NULL	Names of objects that are associated with Database Vault policies
COMMAND_CLAUSE	VARCHAR(100)	NULL	A clause from either the ALTER SYSTEM or ALTER SESSION SQL statement, which was used to create the command rule. For example, you it could list the SET clause for the ALTER SESSION statement. For a full list of possible clause values, see the following topics: • Table 17-2 • Table 17-3
COMMAND_PARAMETER	VARCHAR(128)	NULL	A parameter from the ALTER SYSTEM or ALTER SESSIONcommand rule CLAUSE_NAME setting
COMMAND_EVENT	VARCHAR(128)	NULL	An event that the ALTER SYSTEM or ALTER SESSION command rule defines
COMMAND_COMPONENT	VARCHAR(128)	NULL	A component of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule
COMMAND_ACTION	VARCHAR(128)	NULL	An action of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule
COMMON	VARCHAR(3)	NULL	For a multitenant environment, indicates if the policy objects are local or common. Possible values are: • YES if the policy objects are common • NO if the policy objects are local
INHERITED	VARCHAR(3)	NULL	Shows the inheritance status of the policy object, when the COMMON column output is YES. Values are as follows: • YES means that the policy object was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the policy object is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.

DBA_DV_POLICY_OWNER View

The DBA_DV_POLICY_OWNER data dictionary view lists the owners of Oracle Database Vault policies that were created in the current database instance.

For example:

```
SELECT * FROM DBA_DV_POLICY_OWNER;
```

Output similar to the following appears:

```
POLICY_OWNER                     POLICY_OWNER
-------------------------------- ------------
Oracle System Protection Controls  PSMITH
```

Related Views

- DBA_DV_POLICY View
- DBA_DV_POLICY_OBJECT View

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(128)	NOT NULL	Names of the Oracle Database Vault policies that have been created. See Default Oracle Database Vault Policies for a listing of default policies.
POLICY_OWNER	VARCHAR(128)	NOT NULL	Names of users who have own Database Vault policies

DBA_DV_PREPROCESSOR_AUTH View

The DBA_DV_PREPROCESSOR_AUTH data dictionary view shows users who have been granted authorization to execute preprocessor programs through external tables.

See Using Oracle Database Replay with Oracle Database Vault for more information.

For example:

```
SELECT * FROM DBA_DV_PREPROCESSOR_AUTH WHERE GRANTEE = 'PFITCH';
```

Output similar to the following appears:

```
GRANTEE
-------
PFITCH
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted authorization to execute preprocessor programs

DBA_DV_PROXY_AUTH View

The DBA_DV_PROXY_AUTH data dictionary view lists the proxy users and schemas that were specified by the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure.

This procedure grants a proxy user authorization to proxy other user accounts.

For example:

```
SELECT * FROM DBA_DV_DDL_AUTH WHERE GRANTEE = 'PRESTON';
```

Output similar to the following appears:

```
GRANTEE SCHEMA
------- ------
PRESTON DKENT
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the proxy user
SCHEMA	VARCHAR2(128)	NOT NULL	Name of the schema that is proxied by the GRANTEE user.

See Also:

- AUTHORIZE_PROXY_USER Procedure
- UNAUTHORIZE_PROXY_USER Procedure

DBA_DV_PUB_PRIVS View

The DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in Oracle Database Vault Administrator.

See also Privilege Management - Summary Reports.

For example:

```
SELECT USERNAME, ACCESS_TYPE FROM DBA_DV_PUB_PRIVS WHERE USERNAME = 'OE';
```

Output similar to the following appears:

```
USERNAME    ACCESS_TYPE
----------- -----------------
OE          PUBLIC
```

Related Views

- DBA_DV_USER_PRIVS View
- DBA_DV_USER_PRIVS_ALL View
- DBA_DV_ROLE View

Column	Datatype	Null	Description
USERNAME	VARCHAR(128)	NOT NULL	Database schema in the current database instance.acces
ACCESS_TYPE	VARCHAR(128)	NULL	Access type granted to the user listed in the USERNAME column (for example, PUBLIC).
PRIVILEGE	VARCHAR(40)	NOT NULL	Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(128)	NOT NULL	Owner of the database schema to which the USERNAME user has been granted privileges.
OBJECT_NAME	VARCHAR(128)	NOT NULL	Name of the object within the schema listed in the OWNER column.

DBA_DV_REALM View

The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.

For example:

```
SELECT NAME, AUDIT_OPTIONS, ENABLED, COMMON FROM DBA_DV_REALM
  WHERE AUDIT_OPTIONS = '1';
```

Output similar to the following appears:

```
NAME                          AUDIT_OPTIONS    ENABLED  COMMON
----------------------------- ---------------- -------- ------
Performance Statistics Realm  1                Y        NO
```

Related Views

- DBA_DV_REALM_AUTH View
- DBA_DV_REALM_OBJECT View

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Names of the realms created. SeeDefault Realms for a listing of default realms.
DESCRIPTION	VARCHAR(1024)	NOT NULL	Description of the realm created.
AUDIT_OPTIONS	NUMBER	NOT NULL	Specifies whether auditing is enabled. Possible values are: • 0: No auditing for the realm. • 1: Creates an audit record when a realm violation occurs (for example, when an unauthorized user tries to modify an object that is protected by the realm). • 2: Creates an audit record for authorized activities on objects protected by the realm. • 3: Creates an audit record for both authorized and unauthorized activities on objects protected by the realm.
REALM_TYPE	VARCHAR(9)	NULL	Type of realm: whether it is a regular realm or a mandatory realm. See realm_type in Table 15-9 for a description of the possible values.

Column	Datatype	Null	Description
COMMON	VARCHAR(3)	NOT NULL	For a multitenant environment, indicates whether the realm is local or common. Possible values are: • YES if the realm is common • NO if the realm is local
INHERITED	VARCHAR(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ENABLED	VARCHAR(1)	NOT NULL	Possible values are as follows: • Y indicates that realm checking is enabled • N indicates it is disabled • S indicates the realm is in simulation mode
ID#	NUMBER	NOT NULL	The ID number of the realm, which is automatically generated when the realm is created
ORACLE_SUPPLIED	VARCHAR(3)	NOT NULL	Indicates whether the realm is a default (that is, Oracle-supplied) realm or a user-created command rule. Possible values are: • YES if the realm is a default realm • NO if the realm is a user-created realm
PL_SQL_STACK	VARCHAR(3)	NULL	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

DBA_DV_REALM_AUTH View

The DBA_DV_REALM_AUTH data dictionary view lists database user account or role authorization (GRANTEE) who can access realm objects.

See About Realm Authorization for more information.

For example:

```
SELECT REALM_NAME, GRANTEE, AUTH_RULE_SET_NAME FROM DBA_DV_REALM_AUTH;
```

Output similar to the following appears:

```
REALM_NAME                    GRANTEE  AUTH_RULE_SET_NAME
----------------------------- -------- --------------------
Performance Statistics Realm  SYSADM   Check Conf Access
```

Related Views

- DBA_DV_REALM View
- DBA_DV_REALM_OBJECT View

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(128)	NULL	Name of the realm.
COMMON_REALM	VARCHAR(3)	NULL	For a multitenant environment, indicates whether the realm is local or common. Possible values are: • YES if the realm is common • NO if the realm is local
INHERITED_REALM	VARCHAR(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
GRANTEE	VARCHAR(128)	NOT NULL	User or role name to authorize as owner or participant.
AUTH_RULE_SET_NAME	VARCHAR(128)	NULL	Rule set to check before authorizing. If the rule set evaluates to TRUE, then the authorization is allowed.
AUTH_OPTIONS	VARCHAR(4000)	NULL	Type of realm authorization: either Participant or Owner.
COMMON_AUTH	VARCHAR(3)	NULL	For a multitenant environment, indicates whether the authorization to the common realm is local or common. Possible values are: • YES if the authorization is common • NO if the authorization is local to this PDB

Column	Datatype	Null	Description
INHERITED_AUTH	VARCHAR(3)	NULL	Shows the inheritance status of the realm authorization, when the COMMON_AUTH column output is YES. Values are as follows: • YES means that the realm authorization was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was applied. • NO means that the realm authorization is local, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED_AUTH value NO but a CDB root common command rule will have an INHERITED_AUTH value of YES.

DBA_DV_REALM_OBJECT View

The DBA_DV_REALM_OBJECT data dictionary view lists the database schemas, or subsets of schemas, that are secured by the realms.

See About Realm-Secured Objects for more information.

For example:

```
SELECT REALM_NAME, OWNER, OBJECT_NAME, COMMON_REALM FROM DBA_DV_REALM_OBJECT;
```

Output similar to the following appears:

```
REALM_NAME                    OWNER    OBJECT_NAME COMMON_REALM
----------------------------- -------- ----------- ------------
Performance Statistics Realm OE       ORDERS      NO
```

Related Views

- DBA_DV_REALM View
- DBA_DV_REALM_AUTH View

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(128)	NOT NULL	Name of the realm.
COMMON_REALM	VARCHAR(3)	NOT NULL	Indicates whether this realm is a common realm or a local realm. Possible values are: • YES if the realm is common • NO if the realm is local

Column	Datatype	Null	Description
INHERITED_REALM	VARCHAR(3)	NOT NULL	Shows the inheritance status of the realm when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
OWNER	VARCHAR(128)	NOT NULL	Database schema owner who owns the object.
OBJECT_NAME	VARCHAR(128)	NOT NULL	Name of the object the realm protects.
OBJECT_TYPE	VARCHAR(32)	NOT NULL	Type of object the realm protects, such as a database table, view, index, or role.

DBA_DV_ROLE View

The DBA_DV_ROLE data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.

For example:

```
SELECT ROLE, RULE_NAME FROM DBA_DV_ROLE;
```

Output similar to the following appears:

```
ROLE                 RULE_NAME
-------------------- --------------------
Sector2_APP_MGR      Check App2 Access
Sector2_APP_DBA      Check App2 Access
```

Related Views

- DBA_DV_PUB_PRIVS View
- DBA_DV_USER_PRIVS View
- DBA_DV_USER_PRIVS_ALL View

Column	Datatype	Null	Description
ROLE	VARCHAR(128)	NOT NULL	Name of the secure application role.
RULE_NAME	VARCHAR(128)	NOT NULL	Name of the rule set associated with the secure application role.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the secure application role is enabled. Possible values are: • Y (Yes) if the role is enabled • N (No) if the role is disabled

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	The ID number of the command rule, which is automatically generated when the command rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NOT NULL	Indicates whether the command rule is a default (that is, Oracle-supplied) command rule or a user-created command rule. Possible values are: • YES if the command rule is a default command rule • NO if the command rule is a user-created command rule

DBA_DV_RULE View

The DBA_DV_RULE data dictionary view lists the rules that have been defined.

For example:

```
SELECT NAME, RULE_EXPR FROM DBA_DV_RULE WHERE NAME = 'Maintenance Window';
```

Output similar to the following appears:

```
NAME                RULE_EXP
------------------- ----------------------------------------------
Maintenance Window  TO_CHAR(SYSDATE,'HH24') BETWEEN '10' AND '12'
```

To find the rule sets that use specific rules, query the DBA_DV_RULE_SET_RULE view.

Related Views

- DBA_DV_RULE_SET View
- DBA_DV_RULE_SET_RULE View

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression for the rule.
COMMON	VARCHAR(3)	NOT NULL	For a multitenant environment, indicates whether the rule is local or common. Possible values are: • YES if the rule is common • NO if the rule is local

Column	Datatype	Null	Description
INHERITED	VARCHAR(3)	NULL	Shows the inheritance status of the rule, when the COMMON column output is YES. Values are as follows: • YES means that the rule was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the rule is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ID#	NUMBER	NOT NULL	The ID number of the rule, which is automatically generated when the rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the rule is a default (that is, Oracle-supplied) rule or a user-created rule. Possible values are: • YES if the rule is a default rule • NO if the rule is a user-created rule

DBA_DV_RULE_SET View

The DBA_DV_RULE_SET data dictionary view lists the rules sets that have been created.

For example:

```
SELECT RULE_SET_NAME, HANDLER_OPTIONS, HANDLER FROM DBA_DV_RULE_SET
 WHERE RULE_SET_NAME = 'Maintenance Period';
```

Output similar to the following appears:

```
RULE_SET_NAME       HANDLER_OPTIONS  HANDLER
------------------- ---------------- ----------------------
Maintenance Period                   1 dbavowner.email_alert
```

Related Views

- DBA_DV_RULE View
- DBA_DV_RULE_SET_RULE View

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the rule set.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the rule set has been enabled. Y (Yes) enables the rule set; N (No) disables it.

Column	Datatype	Null	Description
EVAL_OPTIONS_MEANING	VARCHAR(4000)	NULL	For rules sets that contain multiple rules, determines how many rules are evaluated. Possible values are: • All True: All rules in the rule set must evaluate to true for the rule set itself to evaluate to TRUE. • Any True: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to TRUE.
AUDIT_OPTIONS	NUMBER	NOT NULL	Indicates when auditing is used. Possible values are: • 0: No auditing • 1: Audit on failure • 2: Audit on success • 3: Audit on both failure and success
FAIL_OPTIONS_MEANING	VARCHAR(4000)	NULL	Determines when an audit record is created for the rule set. Possible values are: • Do Not Show Error Message. • Show Error Message
FAIL_MESSAGE	VARCHAR(80)	NULL	Error message for failure that is associated with the fail code listed in the FAIL_CODE column.
FAIL_CODE	VARCHAR(10)	NULL	The error message number associated with the message listed in the FAIL_MESSAGE column. Possible values are in the ranges of -20000 to -20999 or 20000 to 20999.
HANDLER_OPTIONS	NUMBER	NOT NULL	Determines how error handling is used. Possible values are: • 0: Disables error handling. • 1: Call handler on rule set failure. • 2: Call handler on rule set success.
HANDLER	VARCHAR(1024)	NULL	Name of the PL/SQL function or procedure that defines the custom event handler logic.
IS_STATIC	VARCHAR2(5)	NULL	Indicates how often the rule set is evaluated during a user session. Possible values are: • TRUE: The rule set is evaluated once, and result of the rule set is reused throughout the user session. • FALSE (default): The rule set is evaluated each time it is accessed during the user session.
COMMON	VARCHAR2(3)	NULL	For a multitenant environment, indicates whether the rule set is local or common. Possible values are: • YES if the rule set is common • NO if the rule set is local

Column	Datatype	Null	Description
INHERITED	VARCHAR2(3)	NULL	Shows the inheritance status of the rule set, when the COMMON column output is YES. Values are as follows: • YES means that the rule set was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the rule set is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ID#	NUMBER)	NOT NULL	The ID number of the rule set, which is automatically generated when the rule set is created
ORACLE_SUPPLIED	VARCHAR2(3)	NULL	Indicates whether the rule set is a default (that is, Oracle-supplied) rule set or a user-created rule set. Possible values are: • YES if the rule set is a default rule set • NO if the rule set is a user-created rule set

DBA_DV_RULE_SET_RULE View

The DBA_DV_RULE_SET_RULE data dictionary view lists rules that are associated with existing rule sets.

For example:

```
SELECT RULE_SET_NAME, RULE_NAME, RULE_EXPR FROM DBA_DV_RULE_SET_RULE
 WHERE RULE_NAME = 'Is Security Officer';
```

Output similar to the following appears:

```
RULE_SET_NAME                RULE_NAME          RULE_EXP
---------------------------- ------------------ --------------------------------
Can Grant VPD Administration Is Security Owner  DBMS_MACUTL.USER_HAS_ROLE_VARCHAR
                                                ('DV_OWNER',
                                                  dvsys.dv_login_user) = 'Y'
```

Related Views

- DBA_DV_RULE View
- DBA_DV_RULE_SET View

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set that contains the rule.

Column	Datatype	Null	Description
RULE_NAME	VARCHAR(128)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression that defines the rule listed in the RULE_NAME column.
ENABLED	VARCHAR(1)		Indicates whether the rule is enabled or disabled. Y (Yes) enables the rule set; N (No) disables it.
RULE_ORDER	NUMBER	NOT NULL	The order in which rules are used within the rule set. Does not apply to this release.
COMMON	VARCHAR(3)	NOT NULL	For a multitenant environment, indicates whether the rule is local or common. Possible values are: • YES if the rule is common • NO if the rule is local
INHERITED	VARCHAR(3)	NOT NULL	Shows the inheritance status of the rule, when the COMMON column output is YES. Values are as follows: • YES means that the rule was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the rule is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.

DBA_DV_STATUS View

The DBA_DV_STATUS data dictionary view shows the status of Oracle Database Vault being enabled and configured.

For example:

```
SELECT * FROM DBA_DV_STATUS;
```

Output similar to the following appears:

```
NAME                 STATUS
-------------------- ------
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     TRUE
```

Related Views

- CDB_DV_STATUS View

Column	Datatype	Null	Description
NAME	VARCHAR2(19)	NOT NULL	Shows either of the following settings: • DV_CONFIGURE_STATUS shows whether Oracle Database Vault has been configured, that is, with the CONFIGURE_DV procedure. • DV_ENABLE_STATUS shows whether Oracle Database Vault has been enabled, that is, with the DBMS_MACADM.ENABLE_DV procedure.
STATUS	VARCHAR2(64)	NOT NULL	TRUE means that Oracle Database Vault is configured or enabled; FALSE means that it is not.

DBA_DV_SIMULATION_LOG View

The DBA_DV_SIMULATION_LOG data dictionary view captures simulation log information for realms and command rules that have had simulation mode enabled.

For example:

```
SELECT USERNAME, COMMAND FROM DBA_DV_SIMULATION_LOG WHERE REALM = 'HR Realm';
```

Output similar to the following appears:

```
USERNAME      COMMAND
------------- -------------------------------------------
PSMITH        SELECT
```

Related Views

- DBA_DV_REALM View for information about simulation mode settings for realms
- DBA_DV_COMMAND_RULE View for information about simulation mode settings for command rules
- DBA_DV_POLICY View for information about simulation mode settings in Oracle Database Vault policies

Column	Datatype	Null	Description
ID	NUMBER	NOT NULL	Simulation log ID
USERNAME	VARCHAR2(128)	NOT NULL	Name of the user whose information is being tracked
COMMAND	VARCHAR2(128)	NOT NULL	Command rule being tracked For a listing of existing command rules, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View.
VIOLATION_TYPE	VARCHAR2(4000)	NULL	Type of violation. See Table 25-2 for more information.
REALM_NAME	DVSYS.DV_OBJ_NAME	NULL	Realm being tracked. For a listing of existing realms, query the DBA_DV_REALM view, described in DBA_DV_REALM View.

Column	Datatype	Null	Description
REALM_TYPE	VARCHAR2(9)	NULL	Type of realm being tracked (for example, mandatory realms).
OBJECT_OWNER	VARCHAR2(128)	NULL	For command rules, the database schema to which the command rule applied
OBJECT_NAME	VARCHAR2(128)	NULL	For command rules, the database object that the command rule protects
OBJECT_TYPE	VARCHAR2(129)	NULL	For command rules, the type of object that is being protected
RULE_SET_NAME	DVSYS.DV_OBJ_NAME	NULL	Rule set being tracked; it is associated with a command rule For a listing of existing rule sets, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View
RETURNCODE	NUMBER	NOT NULL	The Oracle Database ORA error that results if the Database Vault entity was in the enabled state rather than in simulation state
SQLTEXT	VARCHAR2(4000)	NULL	SQL text that the simulation mode captures
AUTHENTICATION_METHOD	VARCHAR2(10)	NULL	Authentication method used. See Default Factors.
CLIENT_IP	VARCHAR2(45)	NULL	The IP address of the machine from which the client is connected
DB_DOMAIN	VARCHAR2(128)	NULL	The domain of the database as specified in the DB_DOMAIN initialization parameter
DATABASE_HOSTNAME	VARCHAR2(128)	NULL	The host name of the computer on which the instance is running
DATABASE_INSTANCE	VARCHAR2(5)	NULL	The instance identification number of the current instance
DATABASE_IP	VARCHAR2(45)	NULL	The IP address of the computer on which the instance is running
DATABASE_NAME	VARCHAR2(128)	NULL	The name of the database as specified in the DB_NAME initialization parameter
DOMAIN	VARCHAR2(4000)	NULL	A named collection of physical, configuration, or implementation-specific factors in the run-time environment. See Default Factors.
ENTERPRISE_IDENTITY	VARCHAR2(1024)	NULL	The enterprise-wide identity for the user. See Default Factors.
IDENTIFICATION_TYPE	VARCHAR2(14)	NULL	The way the user schema was created in the database. See Default Factors.
LANG	VARCHAR2(10)	NULL	The ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter
LANGUAGE	VARCHAR2(100)	NULL	The language and territory your session currently uses, along with the database character set. See Default Factors.

Column	Datatype	Null	Description
MACHINE	VARCHAR2(64)	NULL	The host name for the database client that established the current session. If you must find out whether the computer was used for a client or server session, then you can compare this setting with the Database_Hostname factor to make the determination
NETWORK_PROTOCOL	VARCHAR2(4)	NULL	The network protocol being used for communication, as specified in the `PROTOCOL=protocol` portion of the connect string
PROXY_ENTERPRISE_IDENTITY	VARCHAR2(1024)	NULL	The Oracle Internet Directory DN when the proxy user is an enterprise user
PROXY_USER	VARCHAR2(128)	NULL	The name of the database user who opened the current session on behalf of SESSION_USER
SESSION_USER	VARCHAR2(128)	NULL	The database user name by which the current user is authenticated. This value remains the same throughout the session.
DV$_DBLINK_INFO	VARCHAR2(128)	NULL	Returns the source of a database link session. The string that it returns has this form: SOURCE_GLOBAL_NAME=*dblink_src_global_name*, DBLINK_NAME=*dblink_name*, SOURCE_AUDIT_SESSIONID=*dblink_src_audit_sessionid* In this specification: • *dblink_src_global_name* is the unique global name of the source database • *dblink_name* is the name of the database link on the source database • *dblink_src_audit_sessionid* source database that initiated source database that initiated the connection to the remote database using *dblink_name*
DV$_MODULE	VARCHAR2(64)	NULL	The application name (module) that was set through the DBMS_APPLICATION_INFO PL/SQL package or Oracle Call Interface (OCI).

Column	Datatype	Null	Description
DV$_CLIENT_IDENTIFIER	VARCHAR2(64)	NULL	Returns an identifier that is set by the application through the DBMS_SESSION.SET_IDENTIFIER procedure, the OCI attribute OCI_ATTR_CLIENT_IDENTIFIER, or Oracle Dynamic Monitoring Service (DMS). Various Oracle Database components use this attribute to identify lightweight application users who authenticate as the same database user.
FACTOR_CONTEXT	VARCHAR2(4000)	NULL	An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered
TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	NULL	Time stamp of user action, in UTC (Coordinated Universal Time) time zone
PL_SQL_STACK	CLOB	NULL	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

VIOLATION_TYPE Code Values

Table 25-2 lists the VIOLATION_TYPE code values for the DBA_DV_SIMULATION_LOG view.

Table 25-2 DBA_DV_SIMULATION_LOG VIOLATION_TYPE Code Values

Code	Meaning
1000	Realm violation
1001	Command rule violation
1002	Oracle Data Pump authorization violation
1003	Simulation violation
1004	Oracle Scheduler authorization violation
1005	DDL authorization violation
1006	PARSE_AS_USER violation

Related Topics

- Using Simulation Mode for Logging Realm and Command Rule Activities

DBA_DV_TTS_AUTH View

The DBA_DV_TTS_AUTH data dictionary view lists users who have been granted authorization through the DBMS_MACADM.AUTHORIZE_TTS_USER procedure to perform Oracle Data Pump transportable operations.

See Using Oracle Data Pump with Oracle Database Vault for more information.

For example:

```
SELECT * FROM DBA_DV_TTS_AUTH;
```

Output similar to the following appears:

```
GRANTEE  TSNAME
-------- --------
DB_MGR   HR_TS
```

Related Views

- DBA_DV_DATAPUMP_AUTH View

Column	Datatype	Null	Description
GRANTEE	VARCHAR(128)	NOT NULL	Name of the user who has been granted transportable tablespace authorization
TSNAME	VARCHAR(128)	NOT NULL	Name of the transportable tablespace to which the GRANTEE user has been granted authorization

DBA_DV_USER_PRIVS View

The DBA_DV_USER_PRIVS data dictionary view lists the privileges for a database user account excluding privileges granted through the PUBLIC role.

For example:

```
SELECT USERNAME, ACCESS_TYPE, PRIVILEGE FROM DBA_DV_USER_PRIVS;
```

Output similar to the following appears:

```
USERNAME  ACCESS_TYPE          PRIVILEGE
--------- -------------------- ------------
DVSYS     DV_PUBLIC            EXECUTE
DVOWNER   DV_ADMIN             SELECT
SYS       SELECT_CATALOG_ROLE  SELECT
...
```

Related Views

- DBA_DV_PUB_PRIVS View
- DBA_DV_ROLE View
- DBA_DV_USER_PRIVS_ALL View

Column	Datatype	Null	Description
USERNAME	VARCHAR(128)	NOT NULL	Name of the database schema account in which privileges have been defined.
ACCESS_TYPE	VARCHAR(128)	NULL	Role the database user account listed in the USERNAME column uses to access the database. Oracle Database Vault accounts have direct access.
PRIVILEGE	VARCHAR(40)	NOT NULL	Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(128)	NOT NULL	Name of the database user account.

Column	Datatype	Null	Description
OBJECT_NAME	VARCHAR(128)	NOT NULL	Name of the PL/SQL function or procedure used to define privileges.

DBA_DV_USER_PRIVS_ALL View

The DBA_DV_USER_PRIVS_ALL data dictionary view lists the privileges for a database account including privileges granted through PUBLIC.

For example:

```
SELECT USERNAME, ACCESS_TYPE, PRIVILEGE FROM DBA_DV_USER_PRIVS;
```

Output similar to the following appears:

```
USERNAME            ACCESS_TYPE  PRIVILEGE
------------------- ------------ -----------------
BEA_DVACCTMGR       CONNECT      CREATE_SESSION
LEO_DVOWNER         DIRECT       CREATE PROCEDURE
...
```

Related Views

- DBA_DV_PUB_PRIVS View
- DBA_DV_ROLE View
- DBA_DV_USER_PRIVS View

Column	Datatype	Null	Description
USERNAME	VARCHAR(128)	NULL	Name of the database schema account in which privileges have been defined.
ACCESS_TYPE	VARCHAR(128)	NULL	Role the database user account listed in the USERNAME column uses to access the database. Oracle Database Vault accounts have direct access.
PRIVILEGE	VARCHAR(40)	NULL	Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(128)	NULL	Name of the database user account.
OBJECT_NAME	VARCHAR(128)	NULL	Name of the PL/SQL function or procedure used to define privileges.

DVSYS.DV$CONFIGURATION_AUDIT View

The DVSYS.DV$CONFIGURATION_AUDIT data dictionary view captures DVSYS.AUDIT_TRAIL$ table audit trail records.

It includes records that are related to successful and failed configuration changes made to realms, rules, rule sets, factors, and other Oracle Database Vault policy configuration activities.

For example:

```
SELECT USERNAME, ACTION_NAME FROM DVSYS.DV$CONFIGURATION_AUDIT
WHERE USERNAME = 'PSMITH';
```

Output similar to the following appears:

```
USERNAME   ACTION_NAME
---------- --------------------
PSMITH     Realm Creation Audit
PSMITH     Rule Set Update Audit
```

Related View

- AUDSYS.DV$CONFIGURATION_AUDIT View

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR(255)	NULL	Operating system login user name of the user whose actions were audited
USERNAME	VARCHAR(128)	NULL	Name of the database user whose actions were audited
USERHOST	VARCHAR2(128)	NULL	Client computer name
TERMINAL	VARCHAR2(30)	NULL	Identifier for the user's terminal
TIMESTAMP	DATA	NULL	Date and time of creation of the audit trail entry (in the local database session time zone)
OWNER	VARCHAR2(128)	NULL	Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR2(128)	NULL	Name of the object affected by the action. Expected values are: • ROLE$ • REALM$ • CODE$ • FACTOR$
ACTION	NUMBER	NOT NULL	Numeric action type code. The corresponding name of the action type is in the ACTION_NAME column. See Table 25-3 for a listing of the possible actions.
ACTION_NAME	VARCHAR2(128)	NULL	Name of the action type corresponding to the numeric code in the ACTION column. See Table 25-3 for a listing of the possible actions.
ACTION_OBJECT_ID	NUMBER	NULL	The unique identifier of the record in the table specified under OBJ_NAME
ACTION_OBJECT_NAME	VARCHAR2(128)	NULL	The unique name or natural key of the record in the table specified under OBJ_NAME
ACTION_COMMAND	VARCHAR2(4000)	NULL	The SQL text of the command procedure that was executed that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)	NULL	The labels for all audit options specified in the record that resulted in the audit event being triggered. For example, a factor set operation that is supposed to audit on get failure and get NULL would indicate these two options.

Column	Datatype	Null	Description
RULE_SET_ID	NUMBER	NULL	The unique identifier of the rule set that was executing and caused the audit event to trigger
RULE_SET_NAME	VARCHAR2(128)	NULL	The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER	NULL	Not used
RULE_NAME	VARCHAR2(128)	NULL	Not used
FACTOR_CONTEXT	VARCHAR2(4000)	NULL	An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered
COMMENT_TEXT	VARCHAR2(4000)	NULL	Text comment on the audit trail entry, providing more information about the statement audited
SESSIONID	NUMBER	NOT NULL	Numeric identifier for each Oracle session
ENTRYID	NUMBER	NOT NULL	Same as the value in the ID# column
STATEMENTID	NUMBER	NOT NULL	Numeric identifier for the statement invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
RETURNCODE	NUMBER	NOT NULL	Oracle error code generated by the action. The error code for a statement or procedure invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
EXTENDED_TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	NULL	Time stamp of creation of the audit trail entry (time stamp of user login for entries) in UTC (Coordinated Universal Time) time zone
PROXY_SESSIONID	NUMBER	NULL	Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)	NULL	Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER	NULL	Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)	NULL	Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(128)	NULL	Database login user name of the user whose actions were audited
CREATE_DATE	DATE	NULL	Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(128)	NULL	Same as CREATED_BY column value
UPDATE_DATE	DATE	NULL	Same as UPDATED_BY column value
GRANTEE	VARCHAR2(128)	NULL	User ID of users who have been granted Database Vault-protected roles, realm authorization, command-rule authorization, job scheduler authorization, or Oracle Data Pump authorizations
ENABLED_STATUS	VARCHAR2(1)	NULL	Indicates whether the configuration was enabled

Table 25-3 describes the possible values for the ACTION column of the DVSYS.DV$CONFIGURATION_AUDIT view.

Table 25-3 DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20001	Enable DV enforcement Audit
20002	Disable DV enforcement Audit
20003	Realm Creation Audit
20004	Realm Update Audit
20005	Realm Rename Audit
20006	Realm Deletion Audit
20007	Add Realm Auth Audit
20008	Delete Realm Auth Audit
20009	Update Realm Auth Audit
20010	Add Realm Object Audit
20011	Update Realm Object Audit
20012	Delete Realm Object Audit
20013	Enable Event Audit
20014	Disable Event Audit
20015	Rule Set Creation Audit
20016	Rule Set Update Audit
20017	Rule Set Rename Audit
20018	Rule Set Deletion Audit
20019	Add Rule To Rule Set Audit
20020	Delete Rule From Rule Set Audit
20021	Rule Creation Audit
20022	Rule Update Audit
20023	Rule Rename Audit
20024	Rule Deletion Audit
20025	CommandRule Creation Audit
20026	CommandRule Update Audit
20027	CommandRule Deletion Audit
20028	Authorize Datapump User Audit
20029	Unauthorize Datapump User Audit
20030	Authorize Job User Audit
20031	Unauthorize Job User Audit
20032	Factor_Type Creation Audit
20033	Factor_Type Deletion Audit
20034	Factor_Type Update Audit
20035	Factor_Type Rename Audit

Table 25-3 (Cont.) DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20036	Factor Creation Audit
20037	G_FACTOR_DELETION_AUDIT_CODE
20038	Factor Update Audit
20039	Factor Rename Audit
20040	Add Factor Link Audit
20041	Delete Factor Link Audit
20042	Add Policy Factor Audit
20043	Delete Policy Factor Audit
20044	Create Identity Audit
20045	Delete Identity Audit
20046	Update Identity Audit
20047	Change Identity Factor Audit
20048	Change Identity Value Audit
20049	Create Identity Map Audit
20050	Delete Identity Map Audit
20051	Create Policy Label Audit
20052	Delete Policy Label Audit
20053	Create Mac Policy Audit
20054	Update Mac Policy Audit
20055	Delete Mac Policy Audit
20056	Create Role Audit
20057	Delete Role Audit
20058	Update Role Audit
20059	Rename Role Audit
20060	Create Domain Identity Audit
20061	Drop Domain Identity Audit
20062	Enable Oradebug Audit
20063	Disable Oradebug Audit
20064	Authorize Proxy User Audit
20065	Unauthorize Proxy User Audit
20066	Enable DV Dictionary Accounts Audit
20067	Disable DV Dictionary Accounts Audit
20068	Authorize DDL Audit
20069	Unauthorize DDL Audit
20070	Authorize TTS Audit

Table 25-3 (Cont.) DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20071	Unauthorize TTS Audit
20072	Authorize PREPROCESSOR Audit
20073	Unauthorize PREPROCESSOR Audit
20074	Create Policy Audit
20075	Update Policy Description Audit
20076	Update Policy State Audit
20077	Rename Policy Audit
20078	Drop Policy Audit
20079	Add Realm to Policy Audit
20080	Delete Realm From Policy Audit
20081	Add Command Rule to Policy Audit
20082	Delete Command Rule from Policy Audit
20083	Add Policy Owner Audit
20084	Delete Policy Owner Audit
20085	Authorize Maintenance Audit
20086	Unauthorize Maintenance Audit

DVSYS.DV$ENFORCEMENT_AUDIT View

The DVSYS.DV$ENFORCEMENT_AUDIT data dictionary view provides information about enforcement-related audits from the DVSYS.AUDIT_TRAIL$ table.

It captures user violations on command rules, realms, and factors.

For example:

```
SELECT USERNAME, ACTION_COMMMAND FROM DVSYS.DV$ENFORCEMENT_AUDIT
WHERE OWNER = 'HR';
```

Output similar to the following appears:

```
USERNAME    ACTION_COMMMAND
----------- ------------------------------
PSMITH      CREATE_REALM
```

Related View

- AUDSYS.DV$ENFORCEMENT_AUDIT View

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR(255)	NULL	Operating system login user name of the user whose actions were audited

Column	Datatype	Null	Description
USERNAME	VARCHAR(128)	NULL	Name of the database user whose actions were audited
USERHOST	VARCHAR(255)	NULL	Client computer name
TERMINAL	VARCHAR(255)	NULL	Identifier for the user's terminal
TIMESTAMP	DATE	NULL	Date and time of creation of the audit trail entry (in the local database session time zone)
OWNER	VARCHAR(128)	NULL	Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR(128)	NULL	Name of the object affected by the action. Expected values are: • ROLE$ • REALM$ • CODE$ • FACTOR$
ACTION	NUMBER	NOT NULL	Numeric action type code. The corresponding name of the action type is in the ACTION_NAME column. See Table 25-4 for a listing of the possible actions.
ACTION_NAME	VARCHAR(128)	NULL	Name of the action type corresponding to the numeric code in the ACTION column
ACTION_OBJECT_ID	NUMBER	NULL	The unique identifier of the record in the table specified under OBJ_NAME
ACTION_OBJECT_NAME	VARCHAR(128)	NULL	The unique name or natural key of the record in the table specified under OBJ_NAME
ACTION_COMMAND	VARCHAR2(4000)	NULL	The SQL text of the command procedure that was executed that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)	NULL	The labels for all audit options specified in the record that resulted in the audit event being triggered. For example, a factor set operation that is supposed to audit on get failure and get NULL would indicate these two options.
RULE_SET_ID	NUMBER	NULL	The unique identifier of the rule set that was executing and caused the audit event to trigger
RULE_SET_NAME	VARCHAR(128)	NULL	The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER	NULL	Not used
RULE_NAME	VARCHAR2(128)	NULL	Not used
FACTOR_CONTEXT	VARCHAR2(4000)	NULL	An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered

Column	Datatype	Null	Description
COMMENT_TEXT	VARCHAR2(4000)	NULL	Text comment on the audit trail entry, providing more information about the statement audited
SESSIONID	NUMBER	NOT NULL	Numeric identifier for each Oracle session
ENTRYID	NUMBER	NOT NULL	Same as the value in the ID# column
STATEMENTID	NUMBER	NOT NULL	Numeric identifier for the statement invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
RETURNCODE	NUMBER	NOT NULL	Oracle error code generated by the action. The error code for a statement or procedure invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
EXTENDED_TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	NULL	Time stamp of creation of the audit trail entry (time stamp of user login for entries) in UTC (Coordinated Universal Time) time zone
PROXY_SESSIONID	NUMBER	NULL	Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)	NULL	Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER	NULL	Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)	NULL	Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(128)	NULL	Database login user name of the user whose actions were audited
CREATE_DATE	DATE	NULL	Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(128)	NULL	Same as CREATED_BY column value
UPDATE_DATE	DATE	NULL	Same as UPDATED_BY column value

The following table describes the possible values for the ACTION column of the DVSYS.DV$ENFORCEMENT_AUDIT view.

Table 25-4 DVSYS.DV$ENFORCEMENT_AUDIT View ACTION Values

Action Type Code	Action Name
10000	Factor Evaluation Audit
10001	Factor Assignment Audit
10002	Factor Expression Audit
10003	Realm Violation Audit
10004	Realm Authorization Audit
10005	Command Authorization Audit

Table 25-4 (Cont.) DVSYS.DV$ENFORCEMENT_AUDIT View ACTION Values

Action Type Code	Action Name
10006	Secure Role Audit
10007	Session Initialization Audit
10008	Secure Command Authorization Audit
10009	OLS Session Initialization Audit
10010	OLS Attempt to Upgrade Label Audit
10011	Command Failure Audit

DVSYS.DV$REALM View

The DVSYS.DV$REALM data dictionary view describes settings that were used to create Oracle Database Vault realms, such as which audit options have been assigned or whether the realm is a mandatory realm.

This view also indicates information such as who created and updated the realm, and when the realm was created and updated.

For example:

```
SELECT NAME, CREATED_BY, TYPE FROM DVSYS.DV$REALM WHERE NAME LIKE 'Statistics';
```

Output similar to the following appears:

```
NAME                          CREATED_BY TYPE
----------------------------- ---------- -----
Performance Statistics Realm JGODFREY   2
```

Related Views

- DBA_DV_REALM View

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	ID number of the realm
NAME	VARCHAR2(128)	NOT NULL	Name of the realm
DESCRIPTION	VARCHAR2(1024)	NULL	Description of the realm
AUDIT_OPTIONS	NUMBER	NOT NULL	Audit options set for the realm. See audit_options in Table 15-9 for a description of the possible values.
REALM_TYPE	NUMBER	NULL	Type of realm: whether it is a regular realm or a mandatory realm. See realm_type in Table 15-9 for a description of the possible values.
COMMON	VARCHAR2(3)	NULL	For a multitenant environment, indicates whether the realm is local or common. Possible values are: • YES if the realm is common • NO if the realm is local

Column	Datatype	Null	Description
INHERITED	VARCHAR2(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ENABLED	VARCHAR2(1)	NOT NULL	Whether the realm has been enabled. See enabled in Table 15-9 for a description of the possible values.
VERSION	NUMBER	NULL	Version of Oracle Database Vault in which the realm was created
CREATED_BY	VARCHAR2(128)	NULL	User who created the realm
CREATE_DATE	DATE	NULL	Date on which the realm was created.
UPDATED_BY	VARCHAR2(128)	NULL	User who last updated the realm
UPDATE_DATE	DATE	NULL	Date on which the realm was last updated

DVSYS.POLICY_OWNER_COMMAND_RULE View

The DVSYS.POLICY_OWNER_COMMAND_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the command rules that have been associated with Database Vault policies.

Examples of information that users can find include the command rule name, its associated rule set, and whether it is enabled. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT COMMAND, OBJECT_OWNER, OBJECT_NAME FROM DVSYS.POLICY_OWNER_COMMAND_RULE;
```

Output similar to the following appears:

```
COMMAND       OBJECT_OWNER  OBJECT_NAME
------------- ------------- ------------
SELECT        HR            EMPLOYEES
```

Related Views

- DVSYS.POLICY_OWNER_POLICY View

Column	Datatype	Null	Description
COMMAND	VARCHAR(128)	NOT NULL	Name of the command rule. For a list of default command rules, see Default Command Rules.
CLAUSE_NAME	VARCHAR(100)	NOT NULL	A clause from either the ALTER SYSTEM or ALTER SESSION SQL statement, which was used to create the command rule. For example, you it could list the SET clause for the ALTER SESSION statement. For a full list of possible clause values, see the following topics: • Table 17-2 • Table 17-3
PARAMETER_NAME	VARCHAR(128)	NOT NULL	A parameter from the ALTER SYSTEM or ALTER SESSION command rule CLAUSE_NAME setting
EVENT_NAME	VARCHAR(128)	NOT NULL	An event that the ALTER SYSTEM or ALTER SESSION command rule defines
COMPONENT_NAME	VARCHAR(128)	NOT NULL	A component of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule.
ACTION_NAME	VARCHAR(128)	NOT NULL	An action of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set associated with this command rule.
OBJECT_OWNER	VARCHAR(128)	NOT NULL	The owner of the object that the command rule affects.
OBJECT_NAME	VARCHAR(128)	NOT NULL	The name of the database object the command rule affects (for example, a database table).
ENABLED	VARCHAR(1)	NOT NULL	Y indicates the command rule is enabled; N indicates it is disabled.
PRIVILEGE_SCOPE	NUMBER	NOT NULL	Obsolete column
ID#	NUMBER	NOT NULL	The ID number of the command rule, which is automatically generated when the command rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the command rule is a default (that is, Oracle-supplied) command rule or a user-created command rule. Possible values are: • YES if the command rule is a default command rule • NO if the command rule is a user-created command rule

DVSYS.POLICY_OWNER_POLICY View

The DVSYS.POLICY_OWNER_POLICY data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners.

The columns of the DVSYS.POLICY_OWNER_POLICY view are the same as those in DBA_DV_POLICY. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT POLICY_NAME, STATE FROM DVSYS.POLICY_OWNER_POLICY
  WHERE STATE != 'ENABLED';
```

Output similar to the following appears:

```
POLICY_NAME                      STATE
-------------------------------- --------
HR.EMPLOYEES_pol                 ENABLED
```

Related View

- DBA_DV_POLICY View

DVSYS.POLICY_OWNER_REALM View

The POLICY_OWNER_REALM data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the realms that have been associated with Database Vault policies.

Examples of information that users can find include the realm name, audit options, type, whether it is inherited, and if it is enabled. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT NAME, ENABLED FROM DVSYS.POLICY_OWNER_REALM;
```

Output similar to the following appears:

```
NAME                       ENABLED
-------------------------- --------
HR.EMPLOYEES_realm         S
```

Related Views

- DVSYS.POLICY_OWNER_REALM_AUTH View
- DVSYS.POLICY_OWNER_REALM_OBJECT View

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Names of the realms that have been associated with Database Vault policies. See DBA_DV_REALM View for a full listing of realms.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the realm
AUDIT_OPTIONS	NUMBER	NOT NULL	Audit options set for the realm. See audit_options in Table 15-9 for a description of the possible values.
REALM_TYPE	NUMBER	NULL	Type of realm: whether it is a regular realm or a mandatory realm. See realm_type in Table 15-9 for a description of the possible values.
COMMON_REALM	VARCHAR2(3)	NULL	For a multitenant environment, indicates whether the realm is local or common. Possible values are: • YES if the realm is common • NO if the realm is local

Column	Datatype	Null	Description
INHERITED_REALM	VARCHAR2(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ENABLED	VARCHAR2(1)	NOT NULL	Indicates the enablement status of the realm. Possible values are: • Y for yes (enabled) • N for no (not enabled) • S for simulation mode
ID#	NUMBER	NOT NULL	The ID number of the realm, which is automatically generated when the realm is created
ORACLE_SUPPLIED	VARCHAR(3)	NOT NULL	Indicates whether the realm is a default (that is, Oracle-supplied) realm or a user-created realm. Possible values are: • YES if the realm is a default realm • NO if the realm is a user-created realm

DVSYS.POLICY_OWNER_REALM_AUTH View

The DVSYS.POLICY_OWNER_REALM_AUTH data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the authorization that was granted to realms that have been associated with Database Vault policies.

Examples of the information that users can find are the realm name, grantee, and associated rule set. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT REALM_NAME, INHERITED_REALM FROM DVSYS.POLICY_OWNER_REALM_AUTH;
```

Output similar to the following appears:

```
REALM_NAME                INHERITED
------------------------- --------
HR.EMPLOYEES_realm        NO
```

Related Views

- DVSYS.POLICY_OWNER_REALM View
- DVSYS.POLICY_OWNER_REALM_OBJECT View

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(128)	NOT NULL	Names of the realms that have been associated with Database Vault policies. See DBA_DV_REALM View for a full listing of realms.
COMMON_REALM	VARCHAR2(3)	NULL	For a multitenant environment, indicates whether the realm is local or common.
INHERITED_REALM	VARCHAR2(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
GRANTEE	VARCHAR(128)	NOT NULL	User or role name to authorize as owner or participant.
AUTH_RULE_SET_NAME	VARCHAR(128)	NULL	Rule set to check before authorizing. If the rule set evaluates to TRUE, then the authorization is allowed.
AUTH_OPTIONS	VARCHAR(4000)	NULL	Type of realm authorization: either Participant or Owner.
COMMON_AUTH	VARCHAR(3)	NULL	For a multitenant environment, indicates whether the user who is authorized for this realm is local or common. Possible values are: • YES if the user is a common user • NO if the users is a local user
INHERITED_AUTH	VARCHAR(3)	NULL	Possible values are: • YES • NO

DVSYS.POLICY_OWNER_REALM_OBJECT View

The DVSYS.POLICY_OWNER_REALM_OBJECT data dictionary view enables users to find information about the objects that have been added to realms that are associated with Database Vault policies, such as. Only users who have been granted the DV_POLICY_OWNER role can query this view.

Examples of information that users can find include the realm name, grantee, and associated rule set.

For example:

```
SELECT REALM_NAME, OWNER, OBJECT_NAME, OBJECT_TYPE FROM
DVSYS.POLICY_OWNER_REALM_OBJECT;
```

Output similar to the following appears:

```
REALM_NAME         OWNER  OBJECT_NAME OBJECT_TYPE
------------------ ------ ----------- -----------
HR.EMPLOYEES_realm HR     EMPLOYEES   TABLE
```

Related Views

- DVSYS.POLICY_OWNER_REALM View
- DVSYS.POLICY_OWNER_REALM_AUTH View

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(128)	NOT NULL	Names of the realms that have been associated with Database Vault policies. See DBA_DV_REALM View for a full listing of realms.
COMMON_REALM	VARCHAR2(3)	NULL	For a multitenant environment, indicates whether the realm is local or common.
INHERITED_REALM	VARCHAR2(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
OWNER	VARCHAR(128)	NOT NULL	Database schema owner who owns the object.
OBJECT_NAME	VARCHAR(128)	NOT NULL	Name of the object the realm protects.
OBJECT_TYPE	VARCHAR(32)	NOT NULL	Type of object the realm protects, such as a database table, view, index, or role.

DVSYS.POLICY_OWNER_RULE View

The DVSYS.POLICY_OWNER_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rules that have been associated with rule sets in Database Vault policies, such as the rule name and its expression. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT NAME, RULE_EXPR FROM DVSYS.POLICY_OWNER_RULE WHERE NAME = 'True';
```

Output similar to the following appears:

```
NAME       RULE_EXPR
---------- --------
True       1=1
```

Related Views

- DVSYS.POLICY_OWNER_COMMAND_RULE View
- DVSYS.POLICY_OWNER_RULE_SET View

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression for the rule.
COMMON	VARCHAR(3)	NOT NULL	For a multitenant environment, indicates whether the rule is local or common. Possible values are: • YES if the rule is common • NO if the rule is local
INHERITED	VARCHAR(3)	NULL	Shows the inheritance status of the rule, when the COMMON column output is YES. Values are as follows: • YES means that the rule was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the rule is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ID#	NUMBER	NOT NULL	The ID number of the rule, which is automatically generated when the rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the rule is a default (that is, Oracle-supplied) rule or a user-created rule. Possible values are: • YES if the rule is a default rule • NO if the rule is a user-created rule

DVSYS.POLICY_OWNER_RULE_SET View

The DVSYS.POLICY_OWNER_RULE_SET data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies.

Examples of information that users can find include the rule set name, its handler information, and whether it is enabled. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT RULE_SET_NAME, ENABLED FROM DVSYS.POLICY_OWNER_RULE_SET;
```

Output similar to the following appears:

```
RULE_SET_NAME  ENABLED
-------------- --------
Allow Sessions Y
```

Related Views

- DVSYS.POLICY_OWNER_COMMAND_RULE View
- DVSYS.POLICY_OWNER_RULE View
- DVSYS.POLICY_OWNER_RULE_SET_RULE View

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the rule set.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the rule set has been enabled. Y (Yes) enables the rule set; N (No) disables it.
EVAL_OPTIONS_MEANING	VARCHAR(4000)	NULL	For rules sets that contain multiple rules, determines how many rules are evaluated. Possible values are: • All True: All rules in the rule set must evaluate to true for the rule set itself to evaluate to TRUE. • Any True: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to TRUE.
AUDIT_OPTIONS	NUMBER	NOT NULL	Indicates when auditing is used. Possible values are: • 0: No auditing • 1: Audit on failure • 2: Audit on success • 3: Audit on both failure and success
FAIL_OPTIONS_MEANING	VARCHAR(4000)	NULL	Determines when an audit record is created for the rule set. Possible values are: • Do Not Show Error Message. • Show Error Message
FAIL_MESSAGE	VARCHAR(80)	NULL	Error message for failure that is associated with the fail code listed in the FAIL_CODE column.
FAIL_CODE	VARCHAR(10)	NULL	The error message number associated with the message listed in the FAIL_MESSAGE column. Possible values are in the ranges of -20000 to -20999 or 20000 to 20999.
HANDLER_OPTIONS	NUMBER	NOT NULL	Determines how error handling is used. Possible values are: • 0: Disables error handling. • 1: Call handler on rule set failure. • 2: Call handler on rule set success.
HANDLER	VARCHAR(1024)	NULL	Name of the PL/SQL function or procedure that defines the custom event handler logic.
IS_STATIC	VARCHAR2(5)	NULL	Indicates how often the rule set is evaluated during a user session. Possible values are: • TRUE: The rule set is evaluated once, and result of the rule set is reused throughout the user session. • FALSE (default): The rule set is evaluated each time it is accessed during the user session.
ID#	NUMBER)	NOT NULL	The ID number of the rule set, which is automatically generated when the rule set is created

Column	Datatype	Null	Description
ORACLE_SUPPLIED	VARCHAR2(3)	NULL	Indicates whether the rule set is a default (that is, Oracle-supplied) rule set or a user-created rule set. Possible values are: • YES if the rule set is a default rule set • NO if the rule set is a user-created rule set

DVSYS.POLICY_OWNER_RULE_SET_RULE View

The DVSYS.POLICY_OWNER_RULE_SET_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that contain rules used in Database Vault policies.

Examples of information that users can find include the rule set name and whether it is enabled. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT ENABLED FROM DVSYS.POLICY_OWNER_RULE_SET_RULE WHERE RULE_SET_NAME = 'Can
Maintain Own Account';
```

Output similar to the following appears:

```
ENABLED
----------
Y
```

Related Views

- DVSYS.POLICY_OWNER_COMMAND_RULE View
- DVSYS.POLICY_OWNER_RULE_SET View
- DVSYS.POLICY_OWNER_RULE View

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set that contains the rule.
RULE_NAME	VARCHAR(128)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression that defines the rule listed in the RULE_NAME column.
ENABLED	VARCHAR(1)		Indicates whether the rule is enabled or disabled. Y (Yes) enables the rule set; N (No) disables it.
RULE_ORDER	NUMBER	NOT NULL	The order in which rules are used within the rule set. Does not apply to this release.

AUDSYS.DV$CONFIGURATION_AUDIT View

The AUDSYS.DV$CONFIGURATION_AUDIT view is almost the same as the DVSYS.DV$CONFIGURATION_AUDIT view except that it captures unified audit trail Database Vault audit records.

Related Topics

- DVSYS.DV$CONFIGURATION_AUDIT View
  The `DVSYS.DV$CONFIGURATION_AUDIT` data dictionary view captures `DVSYS.AUDIT_TRAIL$` table audit trail records.

AUDSYS.DV$ENFORCEMENT_AUDIT View

The `AUDSYS.DV$ENFORCEMENT_AUDIT` view is almost the same as the `DVSYS.DV$ENFORCEMENT_AUDIT` view except that it captures unified audit trail Database Vault audit records.

Related Topics

- DVSYS.DV$ENFORCEMENT_AUDIT View
  The `DVSYS.DV$ENFORCEMENT_AUDIT` data dictionary view provides information about enforcement-related audits from the `DVSYS.AUDIT_TRAIL$` table.

26
Monitoring Oracle Database Vault

You can monitor Oracle Database Vault by checking for violations to the Database Vault configurations and by tracking changes to policies.

- About Monitoring Oracle Database Vault
  You can use the Database Vault home page in Oracle Enterprise Manager Cloud Control to monitor a Database Vault-enabled database.
- Monitoring Security Violations and Configuration Changes
  A user who has been granted the appropriate role can use Oracle Database Vault Administrator to monitor security violations and configuration changes.

About Monitoring Oracle Database Vault

You can use the Database Vault home page in Oracle Enterprise Manager Cloud Control to monitor a Database Vault-enabled database.

This feature displays the top five attempted violations and who the top five attempted violators are. The attempted violations cover violations to realms and to command rules. The attempted violators is categorized into users and client hosts. By clicking the **Oracle Database Vault** link under Top 5 Attempted Violations, you can find details such as the type of violation, when it occurred, who the user was, and so on. Similarly, if you click the user link (for example, **SYS**) under Top 5 Attempted Violators, you can find detailed information about each violator, such as the action they performed, the client host name where the action originated, and when the violation occurred. You can manually refresh the data, and restrict the data view, such as within the last 24 hours. This page also shows a table listing all alerts that have been generated.

Before you can view these events, if you have not migrated your database to unified auditing, then you must ensure that the `AUDIT_TRAIL` initialization parameter is set to `DB` or `DB, EXTENDED`. If you have migrated your database to use unified auditing, then you do not need to configure any additional settings. You are ready to check for security violations.

Related Topics

- Oracle Database Vault Reports
  Oracle Database Vault provides reports that track activities, such as the Database Vault configuration settings.

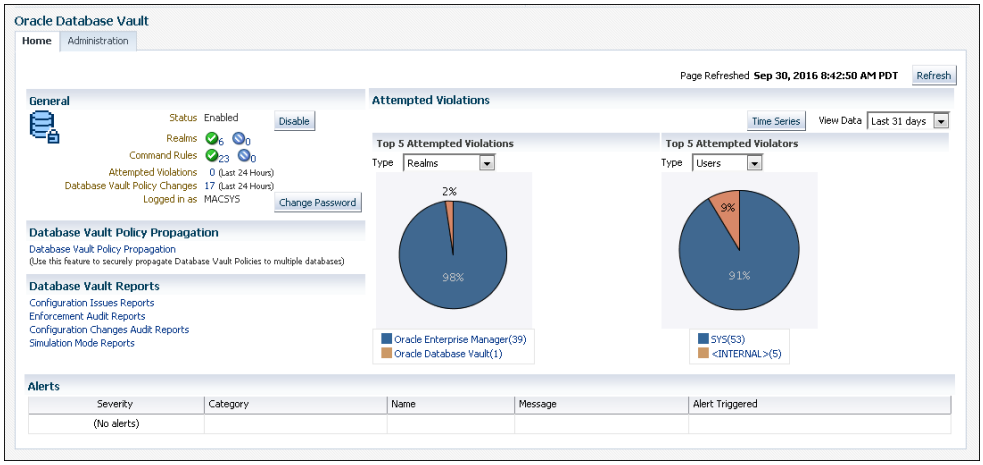
Monitoring Security Violations and Configuration Changes

A user who has been granted the appropriate role can use Oracle Database Vault Administrator to monitor security violations and configuration changes.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.

2. Select the **Home** tab.

   A page similar to the following appears:



3. To find attempted violations for a specific time, such as the last 7 days, select from the menu under the **Time Series** button in the upper right corner.

   You also can change the pie chart to a graph by clicking the **Time Series** button.

4. To find the **Configuration Issues Reports**, **Enforcement Audit Reports**, **Configuration Changes Audit Reports**, and **Simulation Mode Reports**, select the appropriate link under Database Vault reports.

   See Oracle Database Vault Reports for detailed information about the Database Vault reports.

27

Oracle Database Vault Reports

Oracle Database Vault provides reports that track activities, such as the Database Vault configuration settings.

- About the Oracle Database Vault Reports
  Oracle Database Vault provides reports that display security-related information from the database.
- Who Can Run the Oracle Database Vault Reports?
  Users must have the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role before they can run the Oracle Database Vault reports.
- Running the Oracle Database Vault Reports
  A user who has been granted the appropriate roles can run the Oracle Database Vault reports from Database Vault Administrator.
- Oracle Database Vault Configuration Issues Reports
  The configuration issues reports track the settings for command rules, rule sets, realms, and other Oracle Database Vault configurations.
- Oracle Database Vault Auditing Reports
  If you have unified auditing enabled, then the Oracle Database Vault audit reports capture the results of unified audit policies.
- Oracle Database Vault General Security Reports
  The general security reports track information such as object privileges related to `PUBLIC` or privileges granted to a database account or role.

About the Oracle Database Vault Reports

Oracle Database Vault provides reports that display security-related information from the database.

These reports also show custom Oracle Database Vault audit event information. If you have unified auditing enabled, then the reports capture the results of your unified audit policies.

The reports are in two categories:

- **Database Vault Reports.** These reports allow you to check configuration issues with realms, command rules, factors, factor identities, rule sets, and secure application roles. These reports also reveal realm violations, auditing results, and so on.
- **General Security Reports.** These reports allow you to check the status of object privileges, database account system privileges, sensitive objects, privilege management, powerful database accounts and roles, initialization parameters, profiles, account passwords, security audits, and other security vulnerability reports.

Related Topics

- Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control
  From the Database Vault home page, you can find information about violations.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

Who Can Run the Oracle Database Vault Reports?

Users must have the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role before they can run the Oracle Database Vault reports.

Related Topics

- DV_OWNER Database Vault Owner Role
  The `DV_OWNER` role enables you to manage the Oracle Database Vault roles and its configuration.
- DV_ADMIN Database Vault Configuration Administrator Role
  The `DV_ADMIN` role controls the Oracle Database Vault PL/SQL packages.
- DV_SECANALYST Database Vault Security Analyst Role
  The `DV_SECANALYST` role enables users to analyze activities.

Running the Oracle Database Vault Reports

A user who has been granted the appropriate roles can run the Oracle Database Vault reports from Database Vault Administrator.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Home page, under Reports, select **Database Vault Reports**.
3. On the left side, select the category of reports that you want.
   - Database Vault Configuration Issues
   - Database Vault Enforcement Audit Reports
   - Database Vault Configuration Changes
4. In the Reports page, expand the category that contains the report.

   For example, to find the Rule Set Configurations Issues report, you must expand **Database Vault Configuration Issues**.
5. Select the report (for example, **Rule Set Configuration Issues**).

   The report appears in the right pane.
6. Optionally, use the **Search** field to filter the report.

   For example, you can search for reported incidents that involve a specific rule set. The Search field contents vary depending on the report.
7. When you finished viewing the report, click the **OK** button.

Oracle Database Vault Configuration Issues Reports

The configuration issues reports track the settings for command rules, rule sets, realms, and other Oracle Database Vault configurations.

- Command Rule Configuration Issues Report
  The Command Rule Configuration Issues Report displays command rules that have configuration issues.
- Rule Set Configuration Issues Report
  The Rule Set Configuration Issues Report displays Oracle Database Vault rule set configuration issues.
- Realm Authorization Configuration Issues Report
  The Realm Authorization Configuration Issues Report displays Oracle Database Vault realm configuration issues.
- Factor Configuration Issues Report
  The Factor Configuration Issues Report displays Oracle Database Vault factors configuration issues.
- Factor Without Identities Report
  The Factor Without Identities Report displays Oracle Database Vault factors that have no identities configured.
- Identity Configuration Issues Report
  The Identity Configuration Issues Report displays Oracle Database Vault factor identity configuration issues.
- Secure Application Configuration Issues Report
  The Secure Application Configuration Issues Report displays Database Vault secure application role configuration issues.

Command Rule Configuration Issues Report

The Command Rule Configuration Issues Report displays command rules that have configuration issues.

These issues are as follows:

- Rule set for the command rule is disabled.
- Rule set for the command rule is incomplete.
- Object owner for the command rule does not exist. This can happen when the user account for the object has been dropped.

Rule Set Configuration Issues Report

The Rule Set Configuration Issues Report displays Oracle Database Vault rule set configuration issues.

This report tracks when no rules are defined or enabled for a rule set.

Realm Authorization Configuration Issues Report

The Realm Authorization Configuration Issues Report displays Oracle Database Vault realm configuration issues.

These issues are as follows:

- Rule set for a realm authorization is disabled.
- Grantee does not exist for a realm authorization.
- Owner does not exist for a realm-secured object. This can happen when the user account has been dropped.

In most cases, however, these types of issues are caught when you configure the realm and during validation.

Factor Configuration Issues Report

The Factor Configuration Issues Report displays Oracle Database Vault factors configuration issues.

These issues are as follows:

- Rule set for factor assignment is disabled.
- Rule set for factor assignment is incomplete.
- Audit options for the factor are invalid.
- No factor retrieval method or constant exists.
- No subfactors (that is, child factors) are linked to a factor identity.
- No subfactors (child factors) are linked to a label factor.
- Oracle Label Security policy does not exist for the factor.

Factor Without Identities Report

The Factor Without Identities Report displays Oracle Database Vault factors that have no identities configured.

For some factors such as `Background_Job_Id`, this may not be a real problem, but the report can help you determine whether your access control configuration is complete and whether you have accounted for all factor configuration.

Identity Configuration Issues Report

The Identity Configuration Issues Report displays Oracle Database Vault factor identity configuration issues.

These issues are as follows:

- Label identity for the Oracle Label Security label for this identity has been removed and no longer exists.
- No map exists for the identity.

Secure Application Configuration Issues Report

The Secure Application Configuration Issues Report displays Database Vault secure application role configuration issues.

These issues are as follows:

- The database role does not exist. This can happen when the database role has been dropped.
- The rule set for role is disabled.
- The rule set for role is incomplete.

Oracle Database Vault Auditing Reports

If you have unified auditing enabled, then the Oracle Database Vault audit reports capture the results of unified audit policies.

- Realm Audit Report
  The Realm Audit Report shows audit records generated by the realm protection and realm authorization operations.
- Command Rule Audit Report
  The Command Rule Audit Report shows audit records generated by command rule processing operations.
- Factor Audit Report
  The Factor Audit Report shows factors that failed to evaluate or were set to create audit records under various conditions.
- Label Security Integration Audit Report
  The Label Security Integration Audit Report shows audit records the session initialization operation generates and the session label assignment operation of label security.
- Core Database Vault Audit Trail Report
  The Core Database Vault Audit Trail Report shows audit records that the core access security session initialization operation generates.
- Secure Application Role Audit Report
  The Secure Application Role Audit Report shows the audit records that the Oracle Database Vault secure application role-enabling operation generates.

Realm Audit Report

The Realm Audit Report shows audit records generated by the realm protection and realm authorization operations.

You can manage realm authorizations by using rule sets, and then audit the rule set processing results. A realm violation occurs when the database account, performing an action on a realm-protected object, is not authorized to perform that action. Oracle Database Vault audits the violation even if you do not specify any rule sets attached to the realm. When you configure a realm, you can set it to audit instances of realm violations. You can use this information to investigate attempts to break security.

Command Rule Audit Report

The Command Rule Audit Report shows audit records generated by command rule processing operations.

When you configure a command rule, you can set it to audit the rule set processing results.

Factor Audit Report

The Factor Audit Report shows factors that failed to evaluate or were set to create audit records under various conditions.

This report also shows failed attempts to set factors.

You can audit instances where a factor identity cannot be resolved and assigned (such as *No data found* or *Too many rows*). A factor can have an associated rule set that assigns an identity to the factor at run time. When you configure a factor, you can set it to audit the rule set processing results.

Label Security Integration Audit Report

The Label Security Integration Audit Report shows audit records the session initialization operation generates and the session label assignment operation of label security.

You can audit instances where the label security session fails to initialize, and where the label security component prevents a session from setting a label that exceeds the maximum session label.

Core Database Vault Audit Trail Report

The Core Database Vault Audit Trail Report shows audit records that the core access security session initialization operation generates.

You can audit instances where the access security session fails to initialize. It displays the following data:

Data A-R	Data R-U
Account	Rule Set
Command	Timestamp
Instance Number	Rule Set
Object Name	User Host
Return Code	-

Secure Application Role Audit Report

The Secure Application Role Audit Report shows the audit records that the Oracle Database Vault secure application role-enabling operation generates.

Related Topics

- Configuring Secure Application Roles for Oracle Database Vault
  Secure application roles enable you to control how much access users have to an application.

Oracle Database Vault General Security Reports

The general security reports track information such as object privileges related to `PUBLIC` or privileges granted to a database account or role.

- Object Privilege Reports
  The object privilege reports track privileges affected by `PUBLIC`, direct object privileges, and object dependencies.
- Database Account System Privileges Reports
  The database account system privileges reports track activities such as direct, indirect, hierarchical, and `ANY` system privileges.
- Sensitive Objects Reports
  The sensitive objects reports track activities such as grants on the `EXECUTE` privilege on `SYS` schema objects and access to sensitive objects.
- Privilege Management - Summary Reports
  The privilege management summary reports track privilege distribution by grantees, owners, and privileges.
- Powerful Database Accounts and Roles Reports
  The powerful database accounts and roles reports track information about users who have been granted power privileges, such as the `WITH ADMIN` privilege.
- Initialization Parameters and Profiles Reports
  The initialization parameters and profiles reports track database parameters, resource profiles, and system limits.
- Database Account Password Reports
  The database account password reports track default passwords and account statuses of database accounts.
- Security Audit Report: Core Database Audit Report
  The Core Database Audit Report lists database audit trail records..
- Other Security Vulnerability Reports
  Other security vulnerability reports track vulnerabilities that arise with activities such as Java policy grants ir operating system directory objects.

Object Privilege Reports

The object privilege reports track privileges affected by `PUBLIC`, direct object privileges, and object dependencies.

- Object Access By PUBLIC Report
  The Object Access By PUBLIC Report lists all objects whose access has been granted to `PUBLIC`.
- Object Access Not By PUBLIC Report
  The Object Access Not By PUBLIC Report describes the object access used by the database accounts on the Report Parameters page.

- Direct Object Privileges Report
  The Direct Object Privileges Report shows the direct object privileges granted to *nonsystem* database accounts.
- Object Dependencies Report
  The Object Dependencies Report describes dependencies in the database between procedures, packages, functions, package bodies, and triggers.

Object Access By PUBLIC Report

The Object Access By PUBLIC Report lists all objects whose access has been granted to `PUBLIC`.

This report details all the object access the database accounts that you specify on the Report Parameters page, through object grants to `PUBLIC`. On the Reports Parameters page, you can filter the results based on the privilege, the object owner, or the object name.

> **Note:**
>
> This report can be quite large if you choose the defaults.

Object Access Not By PUBLIC Report

The Object Access Not By PUBLIC Report describes the object access used by the database accounts on the Report Parameters page.

It checks the grants to the account directly or through a role, but excluding the grants to `PUBLIC`.

On the Reports Parameters page, you can filter the results based on the privilege, the object owner or the object name.

> **Note:**
>
> This report can be quite large if you choose the defaults.

Direct Object Privileges Report

The Direct Object Privileges Report shows the direct object privileges granted to *nonsystem* database accounts.

The following database accounts are excluded from the report:

Accounts C-O	Accounts P-W
CTXSYS	PUBLIC
DMSYS	SYS
DVSYS	SYSMAN

Accounts C-O	Accounts P-W
LBACSYS	SYSTEM
MDSYS	WKSYS
ORDSYS	WMSYS

Object Dependencies Report

The Object Dependencies Report describes dependencies in the database between procedures, packages, functions, package bodies, and triggers.

The report includes dependencies on views created without any database links.

This report can help you develop a security policy using the principle of least privilege for existing applications. If a database object, such as a `UTL_FILE` package, has privileges granted to `PUBLIC` or some other global role, then you can use the Object Dependencies Report to determine an account that may depend on the object and to determine how the account uses the object. To run the report, enter the database account you are inspecting for dependency and the object it may be dependent on, in the Report Parameters page.

The Report Results page shows the dependent object and object type and the source object name and type. This report shows where the potentially sensitive object is being used. By looking at several accounts, you might be able to see patterns that can help you develop restricted roles. These restricted roles can replace `PUBLIC` grants on widely used sensitive objects.

Database Account System Privileges Reports

The database account system privileges reports track activities such as direct, indirect, hierarchical, and `ANY` system privileges.

- Direct System Privileges By Database Account Report
  The Direct System Privileges By Database Account Report lists system privileges directly granted to the database account selected on the Report Parameters page.
- Direct and Indirect System Privileges By Database Account Report
  The Direct and Indirect System Privileges By Database Account Report displays system privileges for the database account selected on the Report Parameters page.
- Hierarchical System Privileges by Database Account Report
  The Hierarchical System Privileges by Database Account Report shows a hierarchical breakdown of role-based system privileges and direct system privileges.
- ANY System Privileges for Database Accounts Report
  The ANY System Privileges for Database Accounts Report shows `ANY` system privileges granted to the specified database account or role.
- System Privileges By Privilege Report
  The System Privileges By Privilege Report lists database accounts and roles that have the system privilege selected on the Report Parameters page.

Direct System Privileges By Database Account Report

The Direct System Privileges By Database Account Report lists system privileges directly granted to the database account selected on the Report Parameters page.

This report also shows whether a privilege has been granted the `WITH ADMIN` option.

Direct and Indirect System Privileges By Database Account Report

The Direct and Indirect System Privileges By Database Account Report displays system privileges for the database account selected on the Report Parameters page.

The system privileges may have been granted directly or granted through a database role that has the `WITH ADMIN` status.

Hierarchical System Privileges by Database Account Report

The Hierarchical System Privileges by Database Account Report shows a hierarchical breakdown of role-based system privileges and direct system privileges.

These privileges are granted to the database account specified on the Report Parameters page.

ANY System Privileges for Database Accounts Report

The ANY System Privileges for Database Accounts Report shows `ANY` system privileges granted to the specified database account or role.

`ANY` system privileges are very powerful and should be judiciously assigned to accounts and roles.

System Privileges By Privilege Report

The System Privileges By Privilege Report lists database accounts and roles that have the system privilege selected on the Report Parameters page.

Another way to control privileges is to create privilege analysis policies to analyze privilege use.

Sensitive Objects Reports

The sensitive objects reports track activities such as grants on the `EXECUTE` privilege on `SYS` schema objects and access to sensitive objects.

- Execute Privileges to Strong SYS Packages Report
  The Execute Privileges to Strong SYS Packages Report shows database accounts and roles with the `EXECUTE` privilege on powerful system packages.
- Access to Sensitive Objects Report
  The Access to Sensitive Objects Report shows database accounts and roles that have object privileges on system tables or views that have sensitive information.
- Public Execute Privilege To SYS PL/SQL Procedures Report
  The Public Execute Privilege to SYS PL/SQL Procedures Report shows database accounts and roles that have `EXECUTE` privileges on that `SYS` owns.

- Accounts with SYSDBA/SYSOPER Privilege Report
  The Accounts with SYSDBA/SYSOPER Privilege Report displays database accounts that have SYS-privileged connection privileges.

Execute Privileges to Strong SYS Packages Report

The Execute Privileges to Strong SYS Packages Report shows database accounts and roles with the EXECUTE privilege on powerful system packages.

For example, these types of packages can be used to access operating system resources.

The following system PL/SQL packages are included:

Packages D-D	Packages D-U
DBMS_ALERT	DBMS_RANDOM
DBMS_BACKUP_RESTORE	DBMS_REPAIR
DBMS_CAPTURE_ADM	DBMS_REPCAT
DBMS_DDL	DBMS_REPCAT_ADMIN
DBMS_DISTRIBUTED_TRUST_ADMIN	DBMS_RESOURCE_MANAGER
DBMS_FGA	DBMS_RESOURCE_MANAGER_PRIVS
DBMS_JOB	DBMS_RLS
DBMS_LDAP	DBMS_SESSION
DBMS_LOB	DEBUG_EXTPROC
DBMS_LOGMNR	UTL_FILE
DBMS_LOGMNR_D	UTL_HTTP
DBMS_OBFUSCATION_TOOLKIT	UTL_SMTP
DBMS_ORACLE_TRACE_AGENT	UTL_TCP
DBMS_PIPE	-

Access to Sensitive Objects Report

The Access to Sensitive Objects Report shows database accounts and roles that have object privileges on system tables or views that have sensitive information.

This report includes the following system tables and views:

Tables/Views A-O	Tables/Views P-S
ALL_SOURCE	PROFILE$
ALL_USERS	PROXY_ROLE_DATA$
APPROLE$	PROXY_ROLE_INFO$
AUD$	ROLE_ROLE_PRIVS
AUDIT_TRAIL$	SOURCE$
DBA_ROLE_PRIVS	STATS$SQLTEXT
DBA_ROLES	STATS$SQL_SUMMARY

Tables/Views A-O	Tables/Views P-S
DBA_TAB_PRIVS	STREAMS$_PRIVILEGED_USER
DBMS_BACKUP_RESTORE	SYSTEM_PRIVILEGE_MAP
DEFROLE$	TABLE_PRIVILEGE_MAP
FGA_LOG$	TRIGGER$
LINK$	USER$
OBJ$	USER_HISTORY$
OBJAUTH$	USER_TAB_PRIVS
OBJPRIV$	SYSTEM_PRIVILEGE_MAP

Public Execute Privilege To SYS PL/SQL Procedures Report

The Public Execute Privilege to SYS PL/SQL Procedures Report shows database accounts and roles that have `EXECUTE` privileges on that `SYS` owns.

This report can be used to determine which privileges can be revoked from `PUBLIC`, or from other accounts and roles. This reduces vulnerabilities as part of an overall security policy implementation using the principle of least privilege.

Accounts with SYSDBA/SYSOPER Privilege Report

The Accounts with SYSDBA/SYSOPER Privilege Report displays database accounts that have `SYS`-privileged connection privileges.

This report also shows whether the accounts use an external password. However, note that this report does not include operating system users who can become `SYSDBA`.

Privilege Management - Summary Reports

The privilege management summary reports track privilege distribution by grantees, owners, and privileges.

- Privileges Distribution By Grantee Report
  The Privileges Distribution By Grantee Report displays the count of privileges granted to a database account or role.
- Privileges Distribution By Grantee, Owner Report
  The Privileges Distribution By Grantee, Owner Report displays a count of privileges based on the grantee and the owner of the object.
- Privileges Distribution By Grantee, Owner, Privilege Report
  The Privileges Distribution By Grantee, Owner, Privilege Report displays a count of privileges based on the privilege, the grantee, and the object owner.

> **See Also:**
>
> DBA_DV_PUB_PRIVS View to find the values on which the counts listed in these reports are based

Privileges Distribution By Grantee Report

The Privileges Distribution By Grantee Report displays the count of privileges granted to a database account or role.

This report provides insight into accounts and roles that may have powerful privileges.

Privileges Distribution By Grantee, Owner Report

The Privileges Distribution By Grantee, Owner Report displays a count of privileges based on the grantee and the owner of the object.

This report provides insight into accounts or roles that may have powerful privileges. You can use this report if you suspect potential intruders or insider threats are looking for accounts that have powerful privileges as accounts to attack or compromise. If intruders can compromise the account (for example, by guessing the password), they can get more privileges than they already have.

Privileges Distribution By Grantee, Owner, Privilege Report

The Privileges Distribution By Grantee, Owner, Privilege Report displays a count of privileges based on the privilege, the grantee, and the object owner.

This report provides insight into the accounts or roles that may have powerful privileges.

Powerful Database Accounts and Roles Reports

The powerful database accounts and roles reports track information about users who have been granted power privileges, such as the `WITH ADMIN` privilege.

- WITH ADMIN Privilege Grants Report
  The WITH ADMIN Privileges Grants Report shows all database accounts and roles that have been granted privileges with the `WITH ADMIN` clause.
- Accounts With DBA Roles Report
  The Accounts With DBA Roles Report shows all database accounts that have the `DBA` role granted to them.
- Security Policy Exemption Report
  The Security Policy Exemption Report shows database (but not Oracle Database Vault) accounts and roles that have the `EXEMPT ACCESS POLICY` system privilege.
- BECOME USER Report
  The BECOME USER Report shows database accounts roles that have the `BECOME USER` system privilege.
- ALTER SYSTEM or ALTER SESSION Report
  The ALTER SYSTEM or ALTER SESSION Report shows database accounts and roles that have the `ALTER SYSTEM` or `ALTER SESSION` privilege.
- Password History Access Report
  The Password History Access Report shows database accounts that have access to the `USER_HISTORY$` table.

- WITH GRANT Privileges Report
  The WITH GRANT Privileges Report shows database accounts that are granted privileges with the `WITH GRANT` clause.
- Roles/Accounts That Have a Given Role Report
  This report displays the database accounts and roles to which a role has been granted.
- Database Accounts With Catalog Roles Report
  The Database Accounts With Catalog Roles Report displays all database accounts and roles that have the catalog-related roles granted to them.
- AUDIT Privileges Report
  The AUDIT Privileges Report displays all database accounts and roles that have the `AUDIT ANY` or `AUDIT SYSTEM` privilege.
- OS Security Vulnerability Privileges Report
  The OS Security Vulnerability Privileges Report lists database accounts and roles that have privileges to export sensitive information to the operating system.

WITH ADMIN Privilege Grants Report

The WITH ADMIN Privileges Grants Report shows all database accounts and roles that have been granted privileges with the `WITH ADMIN` clause.

This privilege can be misused to give another account more system privileges than required.

Accounts With DBA Roles Report

The Accounts With DBA Roles Report shows all database accounts that have the `DBA` role granted to them.

The `DBA` role is a privileged role that can be misused. It is often granted to a database account to save time and to avoid having to determine the least number of privileges an account really needs. This report can help you to start applying a policy using the principle of least privilege to an existing database.

> **See Also:**
>
> Oracle Database Vault Security Guidelines for guidelines on deciding who should have privileged roles

Security Policy Exemption Report

The Security Policy Exemption Report shows database (but not Oracle Database Vault) accounts and roles that have the `EXEMPT ACCESS POLICY` system privilege.

Accounts that have this privilege can bypass all Virtual Private Database (VPD) policy filters and any Oracle Label Security policies that use Oracle Virtual Private Database indirectly. This is a powerful system privilege that should be granted only if absolutely necessary, as it presents a target to gain access to sensitive information in tables that are protected by Oracle Virtual Private Database or Oracle Label Security. You can

use the auditing policies described in Auditing Oracle Database Vault, to audit the use of this privilege.

BECOME USER Report

The BECOME USER Report shows database accounts roles that have the `BECOME USER` system privilege.

The `BECOME USER` privilege is a very powerful system privilege: it enables the `IMP_FULL_DATABASE` and `EXP_FULL_DATABASE` roles for use with Oracle Data Pump. Accounts that possess this privilege can be misused to get sensitive information or to compromise an application.

ALTER SYSTEM or ALTER SESSION Report

The ALTER SYSTEM or ALTER SESSION Report shows database accounts and roles that have the `ALTER SYSTEM` or `ALTER SESSION` privilege.

Oracle recommends that you restrict these privileges only to those accounts and roles that truly need them (for example, the `SYS` account and the `DV_ADMIN` role). The `ALTER SYSTEM` statement can be used to change the security-related database initialization parameters that are set to recommended values as part of the Oracle Database Vault security strengthening service. Both the `ALTER SYSTEM` and `ALTER SESSION` statements can be used to dump database trace files, potentially containing sensitive configuration information, to the operating system.

> **See Also:**
>
> ALTER SYSTEM and ALTER SESSION Privilege Security Considerations for guidelines on using the `ALTER SYSTEM` and `ALTER SESSION` privileges

Password History Access Report

The Password History Access Report shows database accounts that have access to the `USER_HISTORY$` table.

This table stores hashed passwords that were previously used by each account.

Access to this table can make guessing the existing password for an account easier for someone hacking the database.

WITH GRANT Privileges Report

The WITH GRANT Privileges Report shows database accounts that are granted privileges with the `WITH GRANT` clause.

Remember that `WITH GRANT` is used for object-level privileges: An account that has been granted privileges using the `WITH GRANT` option can be misused to grant object privileges to another account.

Roles/Accounts That Have a Given Role Report

This report displays the database accounts and roles to which a role has been granted.

This report is provided for dependency analysis.

Database Accounts With Catalog Roles Report

The Database Accounts With Catalog Roles Report displays all database accounts and roles that have the catalog-related roles granted to them.

These roles are as follows:

- `DELETE_CATALOG_ROLE`
- `EXECUTE_CATALOG_ROLE`
- `RECOVERY_CATALOG_OWNER`
- `SELECT_CATALOG_ROLE`

These catalog-based roles have a very large number of powerful privileges. They should be granted with caution, much like the `DBA` role, which uses them.

AUDIT Privileges Report

The AUDIT Privileges Report displays all database accounts and roles that have the `AUDIT ANY` or `AUDIT SYSTEM` privilege.

This privilege can be used to disable auditing, which could be used to eliminate the audit trail record of a intruder who has compromised the system. The accounts that have this privilege could be targets for intruders.

OS Security Vulnerability Privileges Report

The OS Security Vulnerability Privileges Report lists database accounts and roles that have privileges to export sensitive information to the operating system.

This report can reveal important vulnerabilities related to the operating system.

Initialization Parameters and Profiles Reports

The initialization parameters and profiles reports track database parameters, resource profiles, and system limits.

- Security Related Database Parameters Report
  The Security Related Database Parameters Report lists database parameters that can cause security vulnerabilities if they not set correctly.
- Resource Profiles Report
  The Resource Profiles Report lists resource profiles that may be allowing unlimited resource consumption.
- System Resource Limits Report
  The System Resource Limits Report provides insight into the current system resource usage by the database.

Security Related Database Parameters Report

The Security Related Database Parameters Report lists database parameters that can cause security vulnerabilities if they not set correctly.

This report can be used to compare the recommended settings with the current state of the database parameter values.

Resource Profiles Report

The Resource Profiles Report lists resource profiles that may be allowing unlimited resource consumption.

Examples of resource profiles are `CPU_PER_SESSION` and `IDLE_TIME`. You should review the profiles that might need a cap on the potential resource usage.

System Resource Limits Report

The System Resource Limits Report provides insight into the current system resource usage by the database.

This report helps determine whether any of these resources are approaching their limits under the existing application load. Resources that show large increases over a short period may point to a denial-of-service (DoS) attack. You might want to reduce the upper limit for the resource to prevent the condition in the future.

Database Account Password Reports

The database account password reports track default passwords and account statuses of database accounts.

- Database Account Default Password Report
  The Database Account Default Password Report lists the database accounts that have default passwords.
- Database Account Status Report
  The Database Account Status Report lists existing database accounts.

Database Account Default Password Report

The Database Account Default Password Report lists the database accounts that have default passwords.

Default passwords are provided during the Oracle Database installation.

You should change the passwords for accounts included in this report to nondefault, complex passwords to help secure the database.

Database Account Status Report

The Database Account Status Report lists existing database accounts.

This report shows the account status for each account, which helps you identify accounts that must be locked. Lock and expiry dates provide information that helps determine whether the account was locked as a result of password aging. If a special

password and resource secure profile is used, then you can identify accounts that are not using them. Accounts not using organizationally defined default tablespaces also can be identified, and the temporary tablespace for accounts can be determined. This report also identifies accounts that use external passwords.

Security Audit Report: Core Database Audit Report

The Core Database Audit Report lists database audit trail records..

This report applies to a non-unified auditing environment.

The Core Database Audit Report returns audit records for the audit policy defined in Auditing Oracle Database Vault, and any auditing records that are generated for audit statements you have defined.

This report only displays audit records that are captured if the database initialization parameter `AUDIT_TRAIL` has been set to `DB` (with unified auditing disabled).

> **See Also:**
>
> *Oracle Database Reference* for more information about the `AUDIT_TRAIL` parameter

Other Security Vulnerability Reports

Other security vulnerability reports track vulnerabilities that arise with activities such as Java policy grants ir operating system directory objects.

- Java Policy Grants Report
  The Java Policy Grants Report shows the Java policy permissions stored in the database.
- OS Directory Objects Report
  The OS Directory Objects Report shows directory objects in the database, their privileges, and whether they are available to `PUBLIC`.
- Objects Dependent on Dynamic SQL Report
  The Objects Dependent on Dynamic SQL Report lists objects that use dynamic SQL.
- Unwrapped PL/SQL Package Bodies Report
  The Unwrapped PL/SQL Package Bodies Report lists PL/SQL package procedures that are not wrapped.
- Username/Password Tables Report
  The Username/Password Tables Report identifies application tables in the database that store user names and password strings.
- Tablespace Quotas Report
  The Tablespace Quotas Report lists database accounts that have quotas on one or more tablespaces.
- Non-Owner Object Trigger Report
  The Non-Owner Object Trigger Report lists non-owner triggers.

Java Policy Grants Report

The Java Policy Grants Report shows the Java policy permissions stored in the database.

This report helps reveal violations to the principle of least privilege. Look for `GRANT`, `READ`, or `WRITE` privileges to `PUBLIC` or other accounts and roles that do not necessarily need the privilege. It is advisable to disable Java loading privileges from `PUBLIC`, if Java is not required in the database.

> **Note:**
>
> Oracle JVM, the Java virtual machine option provided with Oracle Database Vault, must be installed before you can run the Java Policy Grants Report.

OS Directory Objects Report

The OS Directory Objects Report shows directory objects in the database, their privileges, and whether they are available to `PUBLIC`.

Directory objects should exist only for secured operating system (OS) directories, and access to them within the database should be protected. You should never use the root operating system directory on any storage device (for example, `/`), because it allows remote database sessions to look at all files on the device.

Objects Dependent on Dynamic SQL Report

The Objects Dependent on Dynamic SQL Report lists objects that use dynamic SQL.

Potential intruders have a greater chance of using this channel if parameter checking or bind variables are not used. The report helps by narrowing the scope of where to look for problems by pointing out who is using dynamic SQL. Such objects can be a target for a SQL injection attack and must be secured to avoid this type of attack. After determining the objects that use dynamic SQL, do the following:

- Check the privileges that client applications (for example, a Web application) have over the object.
- Check the access granted for the object to `PUBLIC` or a wider account base.
- Validate parameters.
- Use bind variables where possible.

Unwrapped PL/SQL Package Bodies Report

The Unwrapped PL/SQL Package Bodies Report lists PL/SQL package procedures that are not wrapped.

Oracle provides a wrap utility that obfuscates code to the point where it cannot be read in the data dictionary or from the data dictionary views. This helps reduce the ability of an intruder to circumvent data protection by eliminating the ability to read source code that manipulates data.

Username/Password Tables Report

The Username/Password Tables Report identifies application tables in the database that store user names and password strings.

You should examine these tables to determine if the information is encrypted. (Search for column names such as `%USER%NAME%` or `%PASSWORD%`.) If it is not, modify the code and applications using these tables to protect them from being visible to database sessions.

Tablespace Quotas Report

The Tablespace Quotas Report lists database accounts that have quotas on one or more tablespaces.

These tablespaces can become potential targets for denial-of-service (DoS) attacks.

Non-Owner Object Trigger Report

The Non-Owner Object Trigger Report lists non-owner triggers.

These are triggers that are owned by a database account that is different from the account that owns the database object on which the trigger acts.

If the trigger is not part of a trusted database application, then it can *steal* sensitive data, possibly from tables protected through Oracle Label Security or Virtual Private Database (VPD), and place it into an unprotected table for subsequent viewing or export.

A

Auditing Oracle Database Vault

You can audit activities in Oracle Database Vault, such as changes to policy configurations.

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.
- Protection of the Unified Audit Trail in an Oracle Database Vault Environment
  By default, `AUDSYS` schema, which contains the unified audit trail, is not protected by a realm.
- Oracle Database Vault Specific Audit Events
  Oracle Database Vault audit events track activities such as whether an action attempted on a realm was successful.
- Archiving and Purging the Oracle Database Vault Audit Trail
  If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.
- Oracle Database Audit Settings Created for Oracle Database Vault
  When you install Oracle Database Vault, it creates several `AUDIT` settings in the database.

About Auditing in Oracle Database Vault

All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

Optionally, you can audit individual policies that you create for realms, rule sets, and factors. The audit indicates if the user's action succeeded (that is, the policy enabled the user to accomplish a task) or if the user's action failed (the policy was violated). These actions are written to audit logs, whose contents you can find either by querying the appropriate data dictionary views, or running the reports described in Oracle Database Vault Reports.

All configuration changes made to Database Vault are mandatorily audited, including actions of unprivileged users who attempt to modify Database Vault policies.

When you install a new database and configure it to use Oracle Database Vault, then by default it uses a mixed-mode environment, that is, a mixture of unified auditing and pre-migrated auditing. If you have upgraded from previous release, then Database Vault uses the auditing that was available from that release.

Before you migrate to a full unified auditing environment, you can create audit policies as follows:

- **Using the Database Vault APIs:** That is, you use the `DBMS_MACADM` PL/SQL package or the Database Vault pages in Enterprise Manager. In this case, the audit records are written to the Database Vault audit trail, which is stored in the `DVSYS.AUDIT_TRAIL$` table. You can query the `DVSYS.DV$CONFIGURATION_AUDIT` and `DVSYS.DV$ENFORCEMENT_AUDIT` views for these audit records.

- **Using the unified audit policy SQL statements:** These statements are the `CREATE AUDIT POLICY`, `ALTER AUDIT POLICY`, `DROP AUDIT POLICY`, `AUDIT`, and `NO AUDIT` statements. They are written to the unified audit trail, which is captured by the `UNIFIED_AUDIT_TRAIL`, `AUDSYS.DV$CONFIGURATION_AUDIT`, and `AUDSYS.DV$ENFORCEMENT_AUDIT` data dictionary views.

When you migrate to unified auditing, then the auditing features in the Database Vault APIs are no longer effective. You should archive and purge these audit records, as described in Archiving and Purging the Oracle Database Vault Audit Trail. From then on, you can manage Database Vault audit policies through the unified audit policy PL/SQL statements.

Except where noted, the remaining sections of this chapter describe how Database Vault auditing works in a non-unified or mixed mode auditing environment.

See Also:

- *Oracle Database Security Guide* for information about how unified auditing works in Oracle Database Vault and how to create unified audit policies
- Oracle Database Vault Audit Trail Record Format
- The following data dictionary views, which are specific to Database Vault unified auditing:

  DVSYS.DV$CONFIGURATION_AUDIT View

  DVSYS.DV$ENFORCEMENT_AUDIT View
- *Oracle Database Upgrade Guide* to migrate your database to unified auditing

Protection of the Unified Audit Trail in an Oracle Database Vault Environment

By default, `AUDSYS` schema, which contains the unified audit trail, is not protected by a realm.

Related Topics

- Creating a Realm
  To enable realm protection, you create the realm and configure it to include realm-secured objects, roles, and authorizations.

Oracle Database Vault Specific Audit Events

Oracle Database Vault audit events track activities such as whether an action attempted on a realm was successful.

- Oracle Database Vault Policy Audit Events
  Oracle Database Vault uses audit events to track configuration activities.

- Oracle Database Vault Audit Trail Record Format
  If you do not use unified auditing, then Oracle Database Vault writes audit records to the DVSYS.AUDIT_TRAIL$ table.

Oracle Database Vault Policy Audit Events

Oracle Database Vault uses audit events to track configuration activities.

These activities are as follows:

- **Realm Audit.** You can audit both successful and failed actions, based on the auditing option that you set when you created the realm. The exception to this is actions performed by the schema owner.
- **Rule Set Audit.** Audits the rule set processing results. You can audit both successful and failed processing. Realm authorizations can be managed using rule sets. You can audit the rule set processing results. Factor assignments and secure application roles audits can be managed using a rule set.
- **Factor Audit.** You can audit both successful and failed factor processing. For failed factor processing, you can audit on all or any of the following events: Retrieval Error, Retrieval Null, Validation Error, Validation False, Trust Level Null, or Trust Level Less Than Zero.
- **Oracle Label Security Session Initialization Failed.** Audits instances where the Oracle Label Security session fails to initialize.
- **Oracle Label Security Attempt to Upgrade Session Label Failed.** Audits instances where the Oracle Label Security component prevents a session from setting a label that exceeds the maximum session label.

Related Topics

- Setting Audit Options for a Factor
  Under Audit Options, you can generate an audit trail if you are not using a unified audit environment.
- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.
- Oracle Database Vault Reports
  Oracle Database Vault provides reports that track activities, such as the Database Vault configuration settings.

Oracle Database Vault Audit Trail Record Format

If you do not use unified auditing, then Oracle Database Vault writes audit records to the DVSYS.AUDIT_TRAIL$ table.

These audit records are not part of the Oracle Database audit trail, and how auditing is enabled in the database has no effect how Oracle Database Vault collects its audit data in the DVSYS.AUDIT_TRAIL$ table. In fact, even if auditing has been disabled in Oracle Database, then the Oracle Database Vault audit functionality continues to write to the DVSYS.AUDIT_TRAIL$ table.

Users who have been granted the DV_OWNER, DV_ADMIN, DV_SECANALYST or DV_MONITOR role can directly query the DVYS.AUDIT_TRAIL$ table.

Table A-1 describes the format of the audit trail, which you must understand if you plan to create custom reports that use the DVSYS.AUDIT_TRAIL$ table.

Table A-1 Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR2(255)	NULL	Operating system login user name of the user whose actions were audited
USERNAME	VARCHAR2(30)	NULL	Name of the database user whose actions were audited
USERHOST	VARCHAR2(128)	NULL	Client computer name
TERMINAL	VARCHAR2(255)	NULL	Identifier for the user's terminal
TIMESTAMP	DATE	NULL	Date and time of creation of the audit trail entry (in the local database session time zone)
OWNER	VARCHAR2(30)	NULL	Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR2(128)	NULL	Name of the object affected by the action. Expected values are: • ROLE$ • REALM$ • CODE$ • FACTOR$
ACTION	NUMBER	NOT NULL	Numeric action type code. The corresponding name of the action type is in the ACTION_NAME column. See Table 25-3 for a list of the expected ACTION and ACTION_NAME values.
ACTION_NAME	VARCHAR2(128)	NULL	Name of the action type corresponding to the numeric code in the ACTION column
ACTION_OBJECT_ID	NUMBER	NULL	The unique identifier of the record in the table specified under OBJ_NAME. For realms, this field contains a list of comma-separated values of all realm IDs that have the Audit on Failure audit option.
ACTION_OBJECT_NAME	VARCHAR2(128)	NULL	The unique name or natural key of the record in the table specified under OBJ_NAME. For realms, this field contains a list of comma-separated values of all realm names that have the Audit on Failure audit option.
ACTION_COMMAND	VARCHAR2(4000)	NULL	The SQL text of the command procedure that was executed that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)	NULL	The labels for all audit options specified in the record that resulted in the audit event being triggered. For example, a factor set operation that is supposed to audit on get failure and get NULL would indicate these two options.
RULE_SET_ID	NUMBER	NULL	The unique identifier of the rule set that was executing and caused the audit event to trigger

Table A-1 (Cont.) Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR2(30)	NULL	The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER	NULL	Not used
RULE_NAME	VARCHAR2(30)	NULL	Not used
FACTOR_CONTEXT	VARCHAR2(4000)	NULL	An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered
COMMENT_TEXT	VARCHAR2(4000)	NULL	Text comment on the audit trail entry, providing more information about the statement audited
SESSIONID	NUMBER	NOT NULL	Numeric identifier for each Oracle session
ENTRYID	NUMBER	NOT NULL	Same as the value in the ID# column
STATEMENTID	NUMBER	NOT NULL	Numeric identifier for the statement invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
RETURNCODE	NUMBER	NOT NULL	Oracle error code generated by the action. The error code for a statement or procedure invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
EXTENDED_TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	NULL	Time stamp of creation of the audit trail entry (time stamp of user login for entries) in UTC (Coordinated Universal Time) time zone
PROXY_SESSIONID	NUMBER	NULL	Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)	NULL	Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER	NULL	Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)	NULL	Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(30)	NULL	Database login user name of the user whose actions were audited
CREATE_DATE	DATE	NULL	Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(30)	NULL	Same as CREATED_BY column value
UPDATE_DATE	DATE	NULL	Same as UPDATED_BY column value

Archiving and Purging the Oracle Database Vault Audit Trail

If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.

- About Archiving and Purging the Oracle Database Vault Audit Trail
  In a non-unified auditing environment, you can archive the Oracle Database Vault audit trail by exporting the DVSYS.AUDIT_TRAIL$ table to a dump file.

About Archiving and Purging the Oracle Database Vault Audit Trail

In a non-unified auditing environment, you can archive the Oracle Database Vault audit trail by exporting the `DVSYS.AUDIT_TRAIL$` table to a dump file.

You should periodically archive and then purge the audit trail to prevent it from growing too large.

If you choose to migrate to unified auditing, then use this procedure to archive and purge the Database Vault audit trail records after you complete the migration. When unified auditing begins to collect records, then the new records will be available for viewing from the `UNIFIED_AUDIT_TRAIL`, `AUDSYS.DV$CONFIGURATION_AUDIT`, and `AUDSYS.DV$ENFORCEMENT_AUDIT` data dictionary views.

Archiving the Oracle Database Vault Audit Trail

You can use SQL*Plus and Oracle Data Pump to archive the Oracle Database Vault audit trail.

1. Log into the database instance as user `SYS` with the `SYSDBA` administrative privilege.

   ```
   sqlplus sys as sysdba
   Enter password: password
   ```

2. Ensure that the user who will perform archiving has the appropriate privileges.

   For example:

   ```
   GRANT CREATE ANY DIRECTORY, EXP_FULL_DATABASE, UNLIMITED TABLESPACE TO psmith;
   ```

3. Connect as a user who has been granted the `DV_OWNER` or `DV_AUDIT_CLEANUP` role.

   For example:

   ```
   connect ebrown
   Enter password: password
   ```

4. Archive the Oracle Database Vault audit trail into a new table in an appropriate schema.

   For example:

   ```
   CREATE TABLE psmith.dv_audit_trail nologging \
   AS SELECT * FROM DVSYS.AUDIT_TRAIL$;
   ```

5. If the schema is already protected by a realm, then ensure that you or the user performing the export operation has been granted the appropriate authorization to use Oracle Data Pump in a Database Vault environment.

   For example, to authorize user `psmith` to perform Data Pump operations on his own schema:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('PSMITH', 'PSMITH');
   ```

6. Connect as the Data Pump user.

For example:

```
CONNECT psmith
Enter password: password
```

7. Create a directory for the Database Vault audit trail.

```
CREATE DIRECTORY dv_audit_dir AS 'dv_audit_trail_directory';
```

8. Exit SQL*Plus.

```
EXIT
```

9. Using Data Pump, export the Database Vault audit trail into the directory object that you just created.

```
expdp psmith directory=dv_audit_dir tables=psmith.dv_audit_trail \
dumpfile=dv_audit.dmp log=dv_audit_exp.log
```

10. Connect to SQL*Plus as a user who has been granted the `DV_OWNER` role.

```
sqlplus ebrown
Enter password: password
```

11. If you have not done so, then create a realm around the schema that now contains the Database Vault audit trail.

    a. Create the realm. For example:

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'DV Audit Trail Realm',
  description   => 'Realm to protect the DV audit trail',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL +
DBMS_MACUTL.G_REALM_AUDIT_SUCCESS,
  realm_type    => 1);
END;
/
```

    b. Add the schema that contains to audit trail to this realm. For example:

```
BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'DV Audit Trail Realm',
  object_owner => 'psmith',
  object_name  => '%',
  object_type  => '%');
END;
/
```

    c. Authorize a trusted user for this realm.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name  => 'DV Audit Trail Realm',
  grantee     => 'PSMITH',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

See Also:

- Using Oracle Data Pump with Oracle Database Vault for more information about granting users Oracle Data Pump privileges in a Database Vault environment
- *Oracle Database SQL Language Reference* for information about the `CREATE DIRECTORY` statement
- *Oracle Database Utilities* for information about the Oracle Data Pump `expdp` utility
- Oracle Database Vault Realm APIs, for information about the realm-related `DBMS_MACADM` procedures

Purging the Oracle Database Vault Audit Trail

You can purge the Oracle Database Vault audit trail in SQL*Plus.

1. Log into the database instance as a user who has been granted the `DV_OWNER` role or the `DV_AUDIT_CLEANUP` role.

   For example:

   ```
   sqlplus psmith
   Enter password: password
   ```

   Note that the `DV_OWNER` and `DV_AUDIT_CLEANUP` roles do not allow their grantees to truncate the `DVSYS.AUDIT_TRAIL$` system table.

   You can query the `DBA_ROLE_PRIVS` data dictionary view to find the roles that have been granted to a user.

2. Purge the Database Vault audit trail.

   ```
   DELETE FROM DVSYS.AUDIT_TRAIL$;
   ```

Related Topics

- DV_AUDIT_CLEANUP Audit Trail Cleanup Role
  The `DV_AUDIT_CLEANUP` role is used for purge operations.

Oracle Database Audit Settings Created for Oracle Database Vault

When you install Oracle Database Vault, it creates several `AUDIT` settings in the database.

In a non-unified auditing environment, in order for these audit settings to take place, auditing must be enabled in this database. You can check if auditing is enabled by using the `SHOW PARAMETER` command to find the value of the `AUDIT_TRAIL` initialization parameter. By default, auditing is enabled in Oracle Database.

Table A-2 lists the `AUDIT` settings that Oracle Database Vault adds to the database.

Table A-2 Audit Policy Settings Oracle Database Vault Adds to Oracle Database

Audit Setting Type	Audited Statements (BY ACCESS and on Success or Failure Unless Otherwise Noted)
User Audit Settings for DVSYS/DVF User Audit Settings for LBACSYS See Table 14-2 for more information about these accounts. See also these sections for detailed information on the DVSYS and DVF schemas: • DVSYS Schema • DVF Schema	ADMINISTER DATABASE TRIGGER ALTER *object* AUDIT SYSTEM BECOME USER CLUSTER COMMENT CONTEXT CREATE *object* DATABASE LINK DEBUG DIRECTORY DROP *object* EXECUTE LIBRARY (WHENEVER NOT SUCCESSFUL) EXECUTE PROCEDURE (WHENEVER NOT SUCCESSFUL) EXEMPT ACCESS POLICY EXPORT FULL DATABASE GRANT *object* IMPORT FULL DATABASE INDEX MANAGE SCHEDULER MANAGE TABLESPACE MATERIALIZED VIEW (audits both accessing and creating materialized views) SELECT SEQUENCE (WHENEVER NOT SUCCESSFUL) SELECT TABLE (WHENEVER NOT SUCCESSFUL)
Object Audit Settings for DVF	AUDIT PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/TABLE COMMENT TABLE/VIEW DELETE TABLE/VIEW EXECUTE PACKAGE/PROCEDURE/FUNCTION (WHENEVER NOT SUCCESSFUL) GRANT PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/TABLE RENAME PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/VIEW/TABLE SELECT SEQUENCE/TABLE/VIEW (WHENEVER NOT SUCCESSFUL)

Table A-2 (Cont.) Audit Policy Settings Oracle Database Vault Adds to Oracle Database

Audit Setting Type	Audited Statements (BY ACCESS and on Success or Failure Unless Otherwise Noted)
Object Audit Settings for DVSYS	AUDIT PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/TABLE
Object Audit Settings for LBACSYS	COMMENT TABLE/VIEW
	DELETE TABLE/VIEW
	EXECUTE PACKAGE/PROCEDURE/FUNCTION (WHENEVER NOT SUCCESSFUL)
	GRANT PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/TABLE
	INSERT TABLE/VIEW
	RENAME PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/VIEW/TABLE
	SELECT SEQUENCE/TABLE/VIEW (WHENEVER NOT SUCCESSFUL)
	UPDATE TABLE/VIEW

B

Disabling and Enabling Oracle Database Vault

Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

- When You Must Disable Oracle Database Vault
  You may need to disable Oracle Database Vault to perform upgrade tasks or correct erroneous configurations.
- Step 1: Disable Oracle Database Vault
  Be aware that after you disable Oracle Database Vault, Oracle Label Security, which is required to run Database Vault, is still enabled.
- Step 2: Perform the Required Tasks
  At this stage, Oracle Database Vault is disabled and you can perform the required tasks.
- Step 3: Enable Oracle Database Vault
  You can enable Oracle Database Vault and Oracle Label Security from SQL*Plus.

When You Must Disable Oracle Database Vault

You may need to disable Oracle Database Vault to perform upgrade tasks or correct erroneous configurations.

You can reenable Oracle Database Vault after you complete the corrective tasks.

To find the enablement and configuration status of Oracle Database Vault, see Verifying That Database Vault Is Configured and Enabled.

The following situations require you to disable Oracle Database Vault:

- You must install any of the Oracle Database optional products or features, such as Oracle Spatial, or Oracle Multimedia, by using Database Configuration Assistant (DBCA).
- If you did not configure backup `DV_OWNER` and `DV_ACCTMGR` accounts when you registered Oracle Database Vault, and these accounts are inadvertently locked or their passwords forgotten. Note that if your site only has one `DV_OWNER` user and this user has lost his or her password, you will be unable to disable Oracle Database Vault. However, if your site's only `DV_ACCTMGR` user has lost the password, you can disable Database Vault. As a best practice, you should grant the `DV_OWNER` and `DV_ACCTMGR` roles to new or existing user accounts, and use the Database Vault Owner and Account Manager accounts that you created when you registered Database Vault as back-up accounts. (See Backup Oracle Database Vault Accounts for a guideline for avoiding this problem in the future.)
- If you want to register Oracle Internet Directory (OID) using Oracle Database Configuration Assistant (DBCA).

> **Note:**
>
> - Be aware that if you disable Oracle Database Vault, the privileges that were revoked from existing users and roles during installation remain in effect. See Privileges That Are Revoked from Existing Users and Roles for a listing of the revoked privileges.
> - When Oracle Database Vault is disabled, there are some Database Vault features that you can still use.
> - Oracle does not support the deinstallation of Oracle Database Vault.

Step 1: Disable Oracle Database Vault

Be aware that after you disable Oracle Database Vault, Oracle Label Security, which is required to run Database Vault, is still enabled.

1. In SQL*Plus, log in as the Oracle Database Owner (`DV_OWNER`) account, and then disable Oracle Database Vault.

   ```
   sqlplus psmith
   Enter password: password

   EXEC DBMS_MACADM.DISABLE_DV;
   ```

2. In a multitenant environment, connect to the appropriate pluggable database (PDB).

   For example:

   ```
   CONNECT psmith@hrpdb
   Enter password: password
   ```

   To find the available PDBs, query the `DBA_PDBS` data dictionary view. To check the current PDB, run the `show con_name` command.

3. Restart the database.

   ```
   CONNECT SYS AS SYSOPER -- Or, CONNECT SYS@hrpdb AS SYSOPER
   Enter password: password

   SHUTDOWN IMMEDIATE
   STARTUP
   ```

4. For Oracle RAC installations, repeat these steps for each node on which the database is installed.

Step 2: Perform the Required Tasks

At this stage, Oracle Database Vault is disabled and you can perform the required tasks.

You can perform the following types of activities:

- **Use the Oracle Database Vault PL/SQL packages and functions.** For example, to correct a login or `CONNECT` rule set error, use the `DBMS_MACADM` PL/SQL package or the Oracle Database Vault pages in Enterprise Manager Cloud Control. Note

that a CONNECT command rule cannot prevent a user who has the DV_OWNER or DV_ADMIN role from connecting to the database. This enables a Database Vault administrator to correct a misconfigured protection without having to disable Database Vault.

- **Use the SYSTEM or SYS accounts to perform tasks such as creating or changing passwords, or locking and unlocking accounts.** In addition to modifying standard database and administrative user accounts, you can modify passwords and the lock status of any of the Oracle Database Vault-specific accounts, such as users who have been granted the DV_ADMIN or DV_ACCTMGR roles. (See the tip under Oracle Database Vault Accounts Created During Registration for a guideline for avoiding this problem in the future.)
- **Perform the installation or other tasks that require security protections to be disabled.**

Step 3: Enable Oracle Database Vault

You can enable Oracle Database Vault and Oracle Label Security from SQL*Plus.

1. In SQL*Plus, connect as the Oracle Database Owner (DV_OWNER) account, and then enable Database Vault.

   If you are enabling Database Vault from a non-multitenant environment or from a PDB:

   ```
   CONNECT psmith -- Or, CONNECT psmith@hrpdb for a PDB
   Enter password: password

   EXEC DBMS_MACADM.ENABLE_DV;
   ```

   If you are enabling Database Vault from the CDB root, for example:

   ```
   CONNECT c##sec_admin_owen
   Enter password: password
   ```

   Select from one of the following settings:

   ```
   EXEC DBMS_MACADM.ENABLE_DV (strict_mode => 'n');
    -- For regular mode
   EXEC DBMS_MACADM.ENABLE_DV (strict_mode => 'y');
    -- For strict mode
   ```

2. Check if Oracle Label Security is enabled.

   ```
   SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Label Security';
   ```

   Oracle Label security must be enabled before you can use Database Vault. If it is not enabled, then this query returns FALSE.

3. If Oracle Label Security is not enabled, then enable it.

   ```
   CONNECT SYS AS SYSDBA -- Or, CONNECT SYS@hrpdb AS SYSDBA
   Enter password: password

   EXEC LBACSYS.CONFIGURE_OLS;
   EXEC LBACSYS.OLS_ENFORCEMENT.ENABLE_OLS;
   ```

4. Restart the database.

   ```
   CONNECT SYS AS SYSOPER -- Or, CONNECT SYS@hrpdb AS SYSOPER
   Enter password: password
   ```

```
SHUTDOWN IMMEDIATE
STARTUP
```

5. For Oracle RAC installations, repeat these steps for each node on which the database is installed.

C

Postinstallation Oracle Database Vault Procedures

After you register Oracle Database Vault, you can perform specialized tasks, such as configuring it on Oracle Real Application Clusters (Oracle RAC) nodes.

- Configuring Oracle Database Vault on Oracle RAC Nodes
  After you configure Oracle Database Vault for an Oracle Real Application Clusters (Oracle RAC) instance, you must configure each Oracle RAC node.
- Adding Languages to Oracle Database Vault
  By default, Oracle Database Vault loads only the English language tables.
- Deinstalling Oracle Database Vault
  You can remove Oracle Database Vault from an Oracle Database installation, for both to both single-instance and Oracle RAC installations.
- Reinstalling Oracle Database Vault
  You can reinstall Oracle Database Vault by using Database Configuration Assistant and afterward, register Database Vault.

Related Topics

- Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB
  You can convert a standalone Oracle Database Release 12c or later database to a PDB, and then plug this PDB into a CDB.

Configuring Oracle Database Vault on Oracle RAC Nodes

After you configure Oracle Database Vault for an Oracle Real Application Clusters (Oracle RAC) instance, you must configure each Oracle RAC node.

The following procedure assumes that you have a separate Oracle home for each node.

1. Log into the database instance as user `SYS` with the `SYSDBA` administrative privilege.

   ```
   sqlplus sys as sysdba
   Enter password: password
   ```

2. Run the following `ALTER SYSTEM` statements on each Oracle RAC node:

   ```
   ALTER SYSTEM SET AUDIT_SYS_OPERATIONS=TRUE SCOPE=SPFILE; -- For non-unified
   auditing environments
   ALTER SYSTEM SET OS_ROLES=FALSE SCOPE=SPFILE;
   ALTER SYSTEM SET RECYCLEBIN='OFF' SCOPE=SPFILE;
   ALTER SYSTEM SET REMOTE_LOGIN_PASSWORDFILE='EXCLUSIVE' SCOPE=SPFILE;
   ALTER SYSTEM SET SQL92_SECURITY=TRUE SCOPE=SPFILE;
   ```

3. Restart Oracle Database.

   ```
   CONNECT / AS SYSOPER
   Enter password: password
   ```

```
SHUTDOWN IMMEDIATE
STARTUP
```

Adding Languages to Oracle Database Vault

By default, Oracle Database Vault loads only the English language tables.

You can add more languages by running the `DBMS_MACADM.ADD_NLS_DATA` procedure for each new language that you want to add. You can add more than one language to Database Vault.

1. Log into the database instance as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Run the following procedure:

   ```
   EXEC DBMS_MACADM.ADD_NLS_DATA('language');
   ```

   You can specify the *language* setting using any case. For example:

   ```
   EXEC DBMS_MACADM.ADD_NLS_DATA('french');

   EXEC DBMS_MACADM.ADD_NLS_DATA('JAPANESE');
   ```

   Replace *language* with one of the following supported languages:

   - `ENGLISH`
   - `GERMAN`
   - `SPANISH`
   - `FRENCH`
   - `ITALIAN`
   - `JAPANESE`
   - `KOREAN`
   - `BRAZILIAN PORTUGUESE`
   - `SIMPLIFIED CHINESE`
   - `TRADITIONAL CHINESE`

Deinstalling Oracle Database Vault

You can remove Oracle Database Vault from an Oracle Database installation, for both to both single-instance and Oracle RAC installations.

However, you cannot deinstall Database Vault from databases in a multitenant environment. This procedure only applies to legacy, non-CDB Oracle Database environments.

The deinstallation process does not affect the initialization parameter settings, even those settings that were modified during the installation process, nor does it affect Oracle Label Security.

1. Log into the database instance as user `SYS` with the `SYSDBA` administrative privilege, or as user who has the `ALTER SYSTEM` system privilege.

For example:

```
sqlplus psmith -- Or, sqlplus psmith@hrpdb for a pluggable database (PDB)
Enter password: password
```

2. Ensure that the recycle bin is disabled.

```
SHOW PARAMETER RECYCLEBIN
```

3. If the recycle bin is on, then disable it using one of the following statements:

```
ALTER SYSTEM SET RECYCLEBIN = OFF;

ALTER SESSION SET recyclebin = OFF SCOPE = SPFILE;
```

4. Connect as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   For example:

```
CONNECT leo_dvowner -- Or, CONNECT leo_dvowner@hrpdb
Enter password: password
```

5. Run the following procedure to disable Oracle Database Vault:

```
EXEC DBMS_MACADM.DISABLE_DV;
```

6. Connect as `SYS` with the `SYSOPER` privilege and then restart the database.

   For example:

```
CONNECT SYS AS SYSOPER -- Or, CONNECT SYS@hrpdb AS SYSOPER
Enter password: password

SHUTDOWN IMMEDIATE
STARTUP
```

   For Oracle RAC installations, shut down and then restart each database instance as follows:

```
$ srvctl stop database -db db_name
$ srvctl start database -db db_name
```

7. Run the `dvremov.sql` script to remove Oracle Database Vault.

   For example:

```
$ORACLE_HOME/rdbms/admin/dvremov.sql
```

Afterward, you can double-check that Oracle Database Vault is truly deinstalled by logging in to SQL*Plus and entering the following statement:

```
SELECT * FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';
```

If Oracle Database Vault is deinstalled, the following output appears:

```
PARAMETER                      VALUE
------------------------------ ----------------------
Oracle Database Vault          FALSE
```

Reinstalling Oracle Database Vault

You can reinstall Oracle Database Vault by using Database Configuration Assistant and afterward, register Database Vault.

1. Log into the database instance as user `SYS` with the `SYSDBA` administrative privilege.

```
sqlplus sys as sysdba -- Or, sqlplus sys@hrpdb as sysdba
Enter password: password
```

2. Start Database Configuration Assistant (DBCA).
   - **UNIX:** Enter the following command in a shell window:

     ```
     dbca
     ```

   - **Windows:** Use one of the following methods to start DBCA on Windows:
     - Click **Start**, select **Programs** (or **All Programs**), then **Oracle - *HOME_NAME***, then **Configuration and Migration Tools**, and then **Database Configuration Assistant**.
     - Enter the following command at a command prompt:

       ```
       dbca
       ```

3. Use DBCA to configure Database Vault for either a new or an existing database.

   See *Oracle Database 2 Day DBA* for detailed information about creating a database with DBCA.

4. Follow the instructions in Registering Oracle Database Vault with an Oracle Database in a Multitenant Environment to register Database Vault.

D

Oracle Database Vault Security Guidelines

As with all Oracle Database products, you should follow security guidelines to better secure your Oracle Database Vault installation.

- Separation of Duty Guidelines
  Oracle Database Vault is designed to easily implement separation of duty guidelines.
- Managing Oracle Database Administrative Accounts
  Oracle provides guidelines for managing security for administrative accounts such as `SYSTEM` or users who have the `SYSDBA` administrative privilege.
- Accounts and Roles Trusted by Oracle Database Vault
  Oracle Database Vault restricts access to application data from many privileged users and roles in the database.
- Accounts and Roles That Should be Limited to Trusted Individuals
  You should limit powerful accounts and roles only to trusted individuals.
- Guidelines for Using Oracle Database Vault in a Production Environment
  You should follow special guidelines when you run Oracle Database Vault in a production environment.
- Secure Configuration Guidelines
  You should be aware of security considerations for special PL/SQL packages, privileges, and the recycle bin.

Separation of Duty Guidelines

Oracle Database Vault is designed to easily implement separation of duty guidelines.

- How Oracle Database Vault Handles Separation of Duty
  Separation of duty is restricting each user's privileges *only* to the tasks he or she is responsible for, and *no more*.
- Separation of Tasks in an Oracle Database Vault Environment
  Oracle Database Vault defines the several main responsibilities.
- Separation of Duty Matrix for Oracle Database Vault
  Before applying separation of duty, you must understand who performs basic administration tasks in your environment and what these administration tasks are.
- Identification and Documentation of the Tasks of Database Users
  You should document the areas of the tasks that your organization needs.

How Oracle Database Vault Handles Separation of Duty

Separation of duty is restricting each user's privileges *only* to the tasks he or she is responsible for, and *no more*.

You should assign specific categories of privileges to specific users, rather than granting many privileges to one user. Simply put, separation of duty creates accountability for each task that your organization requires.

Separation of duty has taken on increased importance over the past 10 years. For many organizations, separation of duty is a new concept that continues to evolve. Database consolidation, regulatory compliance, and outsourcing are just a few of the drivers for increased separation of duty. Oracle Database Vault separation of duty strengthens security by separating security-related administration from day-to-day DBA operations. You can tailor your Database Vault separation of duty implementation to easily adapt to current and future business requirements. Small organizations, in particular, need flexibility as they attempt to increase their security profile with limited resources.

Separation of Tasks in an Oracle Database Vault Environment

Oracle Database Vault defines the several main responsibilities.

These responsibilities are as follows:

- **Account management.** Account management entails creating, modifying, and dropping user accounts. The `DV_ACCTMGR` role provides these privileges. A primary day-to-day `DV_ACCTMGR` user and a backup `DV_ACCTMGR` user are created during the Oracle Database Vault registration process. As a safety measure, you keep and maintain the backup account in case the primary `DV_ACCTMGR` account owner forgets his or her password or leaves the company.
- **Security administration.** Security administration covers basic security tasks such as creating realms and command rules, setting security policies for database users' access, and authorizing database users for jobs they are allowed to perform. Security administrators also run security audit reports. The `DV_OWNER` and `DV_ADMIN` roles provide these privileges. A primary day-to-day `DV_OWNER` user and a backup `DV_OWNER` user are created during the Oracle Database Vault registration process.

  > **Important:**
  >
  > As a safety measure, you should keep and maintain the backup user account in case the primary `DV_OWNER` account owner forgets his or her password or leaves the company. It is also important that you do not lose access to all of the user accounts that have been granted the `DV_OWNER` role. There is no way to recover the `DV_OWNER` role if you lose access (such as with a lost password or a staff departure) to any account that has the `DV_OWNER` role. If you lose access to the `DV_OWNER` role, then you cannot modify any Database Vault controls or disable Database Vault. To remedy this problem, you can recover the database to the last known point where the database had possession of the Database Vault owner account.

  Optionally, you can consolidate the account management and security administrative responsibilities.
- **Database management.** Database management refers to managing the database system but not accessing business data. It includes the following operations:

- Backup operations require a predefined time to perform the backup using predefined tools.
- Tuning and monitoring operations require ongoing performance monitoring and analysis.
- Patching operations require temporary access only during the time the patching takes place

Oracle strongly recommends that you review database management accounts within the context of separation of duty. Different database administrators may have different responsibilities that require different privileges and roles. Similarly, more experienced database administrators may have more roles and privileges. Instead of granting users the default `DBA` role to users, consider tailoring database administrative roles for specific positions and for seniority in your organization. It is important to use only named accounts for day-to-day activities. Accounts such as `SYS` and accounts that use the `SYSDBA` administrative privilege should be managed with Privileged Account Management (PAM) systems and checked out (and audited) when they are used. You should also manage the backup Oracle Database Vault owner and account management accounts with a PAM system. Within the operating system, you should make the `root` and `oracle` accounts available only through a checkout system, because of the powerful privileges that these accounts have.

You should have separate accounts for database account management, database security administration, and additional named accounts for backup operations. Auditors check for separate database accounts for different responsibilities and being able to track the actions of each account. Less important is the number of users assigned to specific tasks. Remember that Oracle Database Vault audit events are protected and that the Database Vault reports show all attempted violations.

See Also:

- Oracle Database Vault Roles for an in-depth look at how the Oracle Database Vault roles provide for separation of duty
- Oracle Database Vault Accounts Created During Registration for a full list of the default Oracle Database Vault accounts
- Backup Oracle Database Vault Accounts for more information about the importance of backup accounts

Separation of Duty Matrix for Oracle Database Vault

Before applying separation of duty, you must understand who performs basic administration tasks in your environment and what these administration tasks are.

Even if a single database administrator is responsible for managing both new database account provisioning and application patching, it is important to document and plan for each of these tasks. Using separate administration accounts for these types of tasks provides increased accountability and reduces associated risks if and when a single account is compromised by a malicious user. In midsize to large organizations, database administrators typically must perform common administration tasks but they do not need access to business data managed by the application.

Creating a matrix for your separation of duty can help you plan your Database Vault deployment. As needed, you can include additional tasks and associated users to this list. This information should become part of the overall enterprise security documentation for your organization.

Table D-1 shows an example of a separation of duty matrix.

Table D-1 Example Separation of Duty Matrix

User, Process or Application	Account Creation	Database Administration					Security Administrator
		SYSDBA	Backup	Tuning	Patching	Monitoring	
JSMITH	Yes	No	No	No	No	No	No
SHARDY	No	No	No	No	No	No	Yes
PKESTNER	No	No	Yes	No	No	No	No
RTYLER	No	No	No	No	Yes	No	No
SANDERSON	No	No	No	Yes	No	Yes	No
SYSTEM	No	No	No	No	Yes, for EBS patching	No	No
RMAN	No	Yes	Yes	No	No	No	No

In some cases, system management tasks may require temporary access to data through specific tools and programs. When this happens, build provisions for this temporary or emergency access into the Oracle Database Vault rules and rule sets.

Identification and Documentation of the Tasks of Database Users

You should document the areas of the tasks that your organization needs.

These areas are as follows:

- The responsibilities of each administrative user
- The kind of access users need. For example, application owners should have data access and developers need access to development instances only.
- Who must manage the system without accessing business data (for example, users who perform backup, patching, tuning, and monitoring operations)
- The duties of each category of tasks (for example, the files that must be backed up, the applications that require patching, what exactly is monitored). Include the alternate user accounts for each of these tasks.
- The databases and applications that must be protected. This includes Oracle applications, partner applications, and custom applications.
- Who must be authorized to access business data, including the following:
  - Application owners through middle tier processes
  - Business users through an application interface
- Emergency "what if" scenarios, such as how to handle a security breach
- Reporting in a production environment, which should include the following:

  - Who runs the reports
  - Which reports must be run
  - The frequency with which each report is run
  - The users who must receive a copy of each report
- In addition to a separation of duty matrix, the creation of the following matrices:
  - An Oracle Database Vault-specific matrix, which can cover the names and tasks of users who have been granted Database Vault roles
  - An application protection matrix, which can cover the applications to be protected and the types of protections you have put in place.

  Table D-2 shows an example of protections Oracle created for PeopleSoft Applications. SYSADM, PSFTDBA, SYSTEM, and DBA have all been authorized for the appropriate rule sets.

Table D-2 Example Application Protection Maxtrix

Protection Type	SYSADM	PSFTDBA	SYSTEM	DBA
PeopleSoft Realm	Owner	Owner	No Access	No Access
SELECT Command Rule	Not Restricted	Limit PSFTDB Rule Set	No Access	No Access
CONNECT Command Rule	PeopleSoftAccess Rule Set	Not Restricted	Not Restricted	Not Restricted
DROP TABLESPACE Command Rule	Disabled Rule Set	Disabled Rule Set	Disabled Rule Set	Disabled Rule Set

Managing Oracle Database Administrative Accounts

Oracle provides guidelines for managing security for administrative accounts such as SYSTEM or users who have the SYSDBA administrative privilege.

- SYSTEM User Account for General Administrative Uses
  Ideally, the SYSTEM account should only be available as a backup that is checked out and audited while being used.
- SYSTEM Schema for Application Tables
  If you have application tables in the SYSTEM schema, then you should add the SYSTEM account to your realm authorizations for these tables.
- Limitation of the SYSDBA Administrative Privilege
  Limit the SYSDBA administrative privilege to users who must connect using this privilege when absolutely necessary and for applications that still require SYSDBA access.
- Root and Operating System Access to Oracle Database Vault
  For better security, you should carefully monitor root and operating system access to Oracle Database. Vault.

SYSTEM User Account for General Administrative Uses

Ideally, the SYSTEM account should only be available as a backup that is checked out and audited while being used.

Only named accounts should be used for normal database administration tasks - not shared accounts. Doing so increases accountability for administrative actions in the database.

SYSTEM Schema for Application Tables

If you have application tables in the SYSTEM schema, then you should add the SYSTEM account to your realm authorizations for these tables.

This enables these applications to continue to work normally.

You can place restrictions on the SYSTEM account to increase or fine-tune security for these applications. For example, you can create a Database Vault rule set to restrict the SYSTEM user's access to specific IP addresses.

Limitation of the SYSDBA Administrative Privilege

Limit the SYSDBA administrative privilege to users who must connect using this privilege when absolutely necessary and for applications that still require SYSDBA access.

For example, mandatory patching processes require SYSDBA access.

For all other cases, create named database accounts to perform daily database administration. Members of the OSDBA user group are also given the SYSDBA administrative privilege. The database SYS account and accounts with SYSDBA privilege along with the operating system root and oracle accounts should be managed in a Privileged Account Management (PAM) system and checked out only when required.

Related Topics

- Management of SYSDBA Access
  You should avoid using the SYS account and the SYSDBA privilege for normal database maintenance tasks.

Root and Operating System Access to Oracle Database Vault

For better security, you should carefully monitor root and operating system access to Oracle Database. Vault.

Oracle Database Vault prevents highly privileged database users from accessing sensitive data. In addition, if you are using Oracle Database itself, then you can use Transparent Data Encryption to prevent the most highly privileged operating system users from accessing sensitive data. Transparent data encryption enables you to encrypt tablespaces and table columns. This prevents operating system users from browsing through the operating system database files and finding sensitive data. As a best practice, always carefully review and restrict direct access to the operating system.

You should have personalized accounts access the operating system. These personalized accounts should, in the Linux or UNIX environments, login using sudo to

the `oracle` software owner when needed. With `sudo`, you can control which specific command each personalized user can execute. Be sure to prevent the use of the `make`, `relink`, `gdb`, or other commands that could potentially harm the database, for these users. However, if an administrative user must install a patch or perform some other emergency operation, you can enable the `make` and `relink` commands for a limited time, and audit their actions during this period.

> **See Also:**
>
> *Oracle Database Advanced Security Guide* for more information about Transparent Data Encryption

Accounts and Roles Trusted by Oracle Database Vault

Oracle Database Vault restricts access to application data from many privileged users and roles in the database.

However, in some cases, Oracle Database Vaults trusts certain roles and privileges.

Table D-3 lists the trusted roles and privileges that are created when you install Oracle Database Vault.

Table D-3 Trusted Oracle Database Vault Roles and Privileges

Role or Privilege	Status	Description
`DV_ACCTMGR` role	Open	Role created during registration and used for creating new database accounts. As a safety measure, maintain a backup user who has the `DV_ACCTMGR` role and manage this account using a Privileged Account Management (PAM) system. Users who have the `DV_OWNER` role cannot alter this user. Loss of all accounts with the `DV_ACCTMGR` role (such as due to lost passwords or people leaving the organization) is not recoverable. Ensure that a backup `DV_ACCTMGR` account is created for this purpose.
`DV_OWNER` role	Open	Role created during registration and used for managing realms, factors and command rules. This user can add himself or herself to realm authorizations. As a safety measure, maintain a backup user who has the `DV_OWNER` role and manage this account using a Privileged Account Management (PAM) system. Users who have the `DV_OWNER` role cannot alter this user. Loss of all accounts with the `DV_OWNER` role (such as due to lost passwords or people leaving the organization) is not recoverable. Ensure that a backup `DV_OWNER` account is created for this purpose.
`SYSDBA` privilege	Enabled	Privilege created during Oracle Database installation. Required by some Oracle features.

Table D-3 (Cont.) Trusted Oracle Database Vault Roles and Privileges

Role or Privilege	Status	Description
SYSOPER privilege	Enabled	Privilege created during Oracle Database installation. Database startup and shutdown. Granted to SYS only by default.

Related Topics

- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the DV_OWNER and DV_ACCTMGR roles.
- Management of SYSDBA Access
  You should avoid using the SYS account and the SYSDBA privilege for normal database maintenance tasks.
- Management of SYSOPER Access
  By default, Oracle Database limits SYSOPER access to operating system users in the OSOPER group and to the user SYS.

Accounts and Roles That Should be Limited to Trusted Individuals

You should limit powerful accounts and roles only to trusted individuals.

- Management of Users with Root Access to the Operating System
  Users who have root user access have full control over the system.
- Management of the Oracle Software Owner
  Users who have access to a system as the Oracle software owner have control over the Oracle software.
- Management of SYSDBA Access
  You should avoid using the SYS account and the SYSDBA privilege for normal database maintenance tasks.
- Management of SYSOPER Access
  By default, Oracle Database limits SYSOPER access to operating system users in the OSOPER group and to the user SYS.

Management of Users with Root Access to the Operating System

Users who have root user access have full control over the system.

Activities that these users can perform include the following:

- Reading unencrypted files
- Moving and deleting any files
- Starting or stopping any program on the system
- Logging in as any user, including the user who owns the Oracle Database installation

Oracle Database Vault does not provide protection against the operating system root access. Manage the `root` and `oracle` accounts in a Privileged Account Management (PAM) system. Only check these accounts out when they are required for certain tasks. Enhance audit levels when highly privileged operating system accounts are being used, up to an including keystroke capture and video capture.

Management of the Oracle Software Owner

Users who have access to a system as the Oracle software owner have control over the Oracle software.

Activities these users can perform include the following:

- Reading unencrypted database files
- Moving and deleting database files
- Starting or stopping Oracle programs in the system

Oracle Database Vault does not provide protection against the operating system access of the Oracle software owner. Manage the Oracle software owner account in a Privileged Account Management (PAM) system. Only check this account out when it is required for certain tasks. Enhance audit levels when highly privileged operating system accounts are being used, up to an including keystroke capture and video capture.

Management of SYSDBA Access

You should avoid using the `SYS` account and the `SYSDBA` privilege for normal database maintenance tasks.

Instead, use named accounts that have the required system privileges or a specific administrative privilege such as `SYSBACKUP`, `SYSDG`, or `SYSKM`. However, there are cases where the `SYSDBA` privilege is required to perform a patch, upgrade of the database or troubleshoot issues (for example, connecting to a down database).

Because users with the `SYSDBA` privilege could have access to sensitive application data either directly or indirectly (for example, through diagnostics, database upgrades, and patching), use of the `SYSDBA` privilege and accounts must be highly restricted. The list of highly privileged accounts include `SYS` and user accounts with the `SYSDBA` privilege in the database, and the root and oracle accounts in the operating system. Access to highly privileged accounts in the database and the operating system should be on an exception basis and require the user to go through a process to unlock access to these accounts and privileges. Oracle recommends that you manage these accounts with a Privileged Account Management (PAM) system. Only check these accounts out when they are required for certain tasks. Enhance audit levels when highly privileged operating system accounts (`root` and `oracle`) and database accounts (`SYS` account and `SYSDBA` administrative privilege) are being used, up to an including keystroke capture and video capture. When these highly privileged accounts access the database, audit the `SYS` account to monitor their activities.

Management of SYSOPER Access

By default, Oracle Database limits `SYSOPER` access to operating system users in the `OSOPER` group and to the user `SYS`.

This prevents SYSOPER from modifying the Oracle data dictionary directly. The SYSOPER privilege has limited privileges within the database, but individuals with this role can start and shut down the Oracle database. Only grant the SYSOPER privilege to trusted individuals.

Guidelines for Using Oracle Database Vault in a Production Environment

You should follow special guidelines when you run Oracle Database Vault in a production environment.

These guidelines are as follows:

- Run a full test of your applications to ensure that the Database Vault policies you have created are working as expected
- Monitor the performance of your applications, and if necessary, tune your rule expressions
- Assign responsibilities to the appropriate production support and security groups, as follows:
  - Assign security responsibilities to the database security administrator.
  - Assign account management to the database account manager.
  - Assign resource management tasks to database administrators.
- Back up your Database Vault API scripts to a secure server.

Secure Configuration Guidelines

You should be aware of security considerations for special PL/SQL packages, privileges, and the recycle bin.

- General Secure Configuration Guidelines
  General secure configuration guidelines involved patches and revoke operations.
- UTL_FILE and DBMS_FILE_TRANSFER Package Security Considerations
  You should carefully restrict access to the UTL_FILE and DBMS_FILE_TRANSFER PL/SQL packages.
- CREATE ANY JOB Privilege Security Considerations
  The CREATE ANY JOB privilege has been revoked from the DBA and the SCHEDULER_ADMIN roles.
- CREATE EXTERNAL JOB Privilege Security Considerations
  The CREATE EXTERNAL JOB privilege was introduced in Oracle Database 10*g* release 2 (10.2).
- LogMiner Package Security Considerations
  The role EXECUTE_CATALOG_ROLE no longer has the EXECUTE privilege granted by default on the several LogMiner packages.
- ALTER SYSTEM and ALTER SESSION Privilege Security Considerations
  You should be aware of ways to secure the powerful ALTER SYSTEM and ALTER SESSION system privileges.

General Secure Configuration Guidelines

General secure configuration guidelines involved patches and revoke operations.

- Installing patches and new applications might re-grant some of the privileges that Oracle recommends that you revoke in this section. Check these privileges after you install patches and new applications to verify that they are still revoked.
- When you revoke `EXECUTE` privileges on packages, ensure that you grant `EXECUTE` on the packages to the owner, check the package dependencies, and recompile any invalid packages after the revoke.

  To find users who have access to the package, log into the database instance as a named database administrator and issue the following query.

  ```
  SELECT * FROM DBA_TAB_PRIVS WHERE TABLE_NAME = package_name;
  ```

  package_name is the name of the package you are looking for.

  To find the users, packages, procedures, and functions that are dependent on the package, issue this query:

  ```
  SELECT OWNER, NAME, TYPE  FROM ALL_DEPENDENCIES
  WHERE REFERENCED_NAME = package_name;
  ```

  Note that these two queries do not identify references to packages made through dynamic SQL.

UTL_FILE and DBMS_FILE_TRANSFER Package Security Considerations

You should carefully restrict access to the `UTL_FILE` and `DBMS_FILE_TRANSFER` PL/SQL packages.

- About Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages
  The `UTL_FILE` package is owned by `SYS` and granted to `PUBLIC`.
- Securing Access to the DBMS_FILE_TRANFER Package
  You can secure access to the `DBMS_FILE_TRANSFER` PL/SQLpackage in a variety of ways.
- Example: Creating a Command Rule to Deny Access to CREATE DATABASE LINK
  The `DBMS_MACADM.CREATE_COMMAND_RULE` enables you to create command rules to deny access to the `CREATE DATABASE LINK` SQL statement.
- Example: Creating a Command Rule to Enable Access to CREATE DATABASE LINK
  The `DBMS_MACADM.UPDATE_COMMAND_RULE` procedure can be used to modify an existing command rule.
- Example: Command Rules to Disable and Enable Access to CREATE DIRECTORY

About Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages

The UTL_FILE package is owned by SYS and granted to PUBLIC.

However, a user must have access to the directory object to manipulate the files in that operating system directory.

The DBMS_FILE_TRANSFER package is owned by SYS and granted to the EXECUTE_CATALOG_ROLE. Users with EXECUTE access on this package can move files from one location to another on the same file system. They also can move files between database instances, including databases on remote systems.

> **See Also:**
>
> *Oracle Database PL/SQL Packages and Types Reference* for information about configuring the UTL_FILE package securely

Securing Access to the DBMS_FILE_TRANFER Package

You can secure access to the DBMS_FILE_TRANSFER PL/SQLpackage in a variety of ways.

- Use any of the following methods to secure the DBMS_FILE_TRANSFER PL/SQLpackage:
  - Revoke the EXECUTE privilege from the DBMS_FILE_TRANSFER package and grant the EXECUTE privilege only to trusted users who need it.
  - Create command rules to control the CREATE DATABASE LINK and CREATE DIRECTORY SQL statements. See Creating a Command Rule for information on creating command rules by using Oracle Database Vault Administrator.
  - Create Oracle Database Vault command rules to limit and enable access to the CREATE DATABASE LINK and CREATE DIRECTORY statements, which are used to establish connections to remote databases.

> **See Also:**
>
> The following sections for examples of command rules that you can create to protect use of the CREATE DATABASE LINK statement:
>
> - Example: Creating a Command Rule to Deny Access to CREATE DATABASE LINK
> - Example: Creating a Command Rule to Enable Access to CREATE DATABASE LINK
> - Example: Command Rules to Disable and Enable Access to CREATE DIRECTORY

Example: Creating a Command Rule to Deny Access to CREATE DATABASE LINK

The `DBMS_MACADM.CREATE_COMMAND_RULE` enables you to create command rules to deny access to the `CREATE DATABASE LINK` SQL statement.

Example D-1 shows how to create a command rule to deny access to the `CREATE DATABASE LINK` privilege.

Example D-1 Creating a Command Rule to Deny Access to CREATE DATABASE LINK

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE (
  command       => 'CREATE DATABASE LINK',
  rule_set_name => 'Disabled',
  object_owner  => '%',
  object_name   => '%',
  enabled       => DBMS_MACUTL.G_YES);
  END;
  /
COMMIT;
```

Example: Creating a Command Rule to Enable Access to CREATE DATABASE LINK

The `DBMS_MACADM.UPDATE_COMMAND_RULE` procedure can be used to modify an existing command rule.

Example D-2 shows how to create a command rule that enables access to the `CREATE DATABASE LINK` privilege.

When a valid user must use the `CREATE DATABASE LINK` statement, the Oracle Database Vault owner can reenable it from Oracle Database Vault Administrator or issue the following commands in SQL*Plus.

Example D-2 Creating a Command Rule to Enable Access to CREATE DATABASE LINK

```
BEGIN
 DBMS_MACADM.UPDATE_COMMAND_RULE (
  command       => 'CREATE DATABASE LINK',
  rule_set_name => 'Enabled',
  object_owner  => '%',
  object_name   => '%',
  enabled       => DBMS_MACUTL.G_YES);
 END;
 /
COMMIT;
```

Example: Command Rules to Disable and Enable Access to CREATE DIRECTORY

Example D-3 shows command rules that disable and enable access to `CREATE DIRECTORY`.

Example D-3 Command Rules to Disable and Enable Access to CREATE DIRECTORY

```
-- Disable access to CREATE DIRECTORY
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE (
  command       => 'CREATE DIRECTORY',
  rule_set_name => 'Disabled',
  object_owner  => '%',
  object_name   => '%',
  enabled       => dbms_macutl.g_yes);
 END;
  /
COMMIT;

-- Enable access to CREATE DIRECTORY
BEGIN
 dbms_macadm.update_command_rule (
  command       => 'CREATE DIRECTORY',
  rule_set_name => 'Enabled',
  object_owner  => '%',
  object_name   => '%',
  enabled       => dbms_macutl.g_yes);
 END;
 /
COMMIT;
```

CREATE ANY JOB Privilege Security Considerations

The `CREATE ANY JOB` privilege has been revoked from the `DBA` and the `SCHEDULER_ADMIN` roles.

Ensure that this change does not affect your applications.

Related Topics

- Using Oracle Scheduler with Oracle Database Vault
  Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific authorization.

CREATE EXTERNAL JOB Privilege Security Considerations

The `CREATE EXTERNAL JOB` privilege was introduced in Oracle Database 10*g* release 2 (10.2).

This privilege is required for database users who want to execute jobs that run on the operating system outside the database. By default, the `CREATE EXTERNAL JOB` privilege is granted to all users who have been granted the `CREATE JOB` privilege. For greater security, revoke this privilege from users who do not need it and then grant it only to those users who do need it.

LogMiner Package Security Considerations

The role `EXECUTE_CATALOG_ROLE` no longer has the `EXECUTE` privilege granted by default on the several LogMiner packages.

These packages are as follows:

- DBMS_LOGMNR
- DBMS_LOGMNR_D
- DBMS_LOGMNR_LOGREP_DICT
- DBMS_LOGMNR_SESSION

You should ensure that this change does not affect your applications.

ALTER SYSTEM and ALTER SESSION Privilege Security Considerations

You should be aware of ways to secure the powerful ALTER SYSTEM and ALTER SESSION system privileges.

- About ALTER SYSTEM and ALTER SESSION Privilege Security Considerations
  Be aware that trace and debug commands have the potential to show Oracle database memory information.
- Example: Adding Rules to the Existing ALTER SYSTEM Command Rule
  You can create a rule that prevents users with the ALTER SYSTEM privilege from issuing ALTER SYSTEM statements.

About ALTER SYSTEM and ALTER SESSION Privilege Security Considerations

Be aware that trace and debug commands have the potential to show Oracle database memory information.

Oracle Database Vault does not protect against these commands. To help secure the Oracle database memory information, Oracle recommends that you strictly control access to the ALTER SYSTEM and ALTER SESSION privileges. These privileges can be granted by the user SYS when connected as SYSDBA and by any user granted the DBA role.

Oracle also recommends that you add rules to the existing command rule for ALTER SYSTEM statement. You can use Oracle Database Vault Administrator to create a rule and add it to a rule set. You should grant the ALTER SESSION privilege only to trusted users. (For example, the ALTER SESSION statement can enable tracing.)

Example: Adding Rules to the Existing ALTER SYSTEM Command Rule

You can create a rule that prevents users with the ALTER SYSTEM privilege from issuing ALTER SYSTEM statements.

Example D-4 shows how to create a rule that prevents users with ALTER SYSTEM privilege from issuing the ALTER SYSTEM DUMP statement. Log into the database instance as the Oracle Database Vault Owner when you create this command rule.

Alternatively, you can use Oracle Database Vault Administrator to create and add this rule to the rule set. See Creating a Rule to Add to a Rule Set for more information.

Example D-4 Adding Rules to the Existing ALTER SYSTEM Command Rule

```
CONNECT bea_dvacctmgr
Enter password: password
```

```
BEGIN
 DBMS_MACADM.CREATE_RULE('NO_SYSTEM_DUMP',
 '(INSTR(UPPER(DV_SQL_TEXT),''DUMP'') = 0)');
 END;
/
EXEC DBMS_MACADM.ADD_RULE_TO_RULE_SET
  ('Allow Fine Grained Control of System Parameters','NO_SYSTEM_DUMP');

COMMIT;
```

E

Troubleshooting Oracle Database Vault

You can troubleshoot Oracle Database Vault by using tools such as trace files or checking certain Oracle Database Vault reports.

- Using Trace Files to Diagnose Oracle Database Vault Events
  Trace files, which the database generates, capture important information to help you debug errors.
- General Diagnostic Tips
  Oracle provides general tips for diagnosing problems in realms, factors, and rule sets.
- Configuration Problems with Oracle Database Vault Components
  Oracle Database Vault provides reports to check configuration problems with realms, command rules, factors, rule sets, or secure application roles.
- Resetting Oracle Database Vault Account Passwords
  Backup accounts can help you reset lost passwords for users who have been granted the `DV_OWNER` and `DV_ACCTMGR` roles.

Using Trace Files to Diagnose Oracle Database Vault Events

Trace files, which the database generates, capture important information to help you debug errors.

- About Using Trace Files to Diagnose Oracle Database Vault Events
  You can monitor the Oracle Database Vault database instance for server and background process events by enabling and checking the database instance trace files.
- Types of Oracle Database Vault Trace Events That You Can and Cannot Track
  You can use trace files to track a variety of Oracle Database Vault activities.
- Levels of Oracle Database Vault Trace Events
  You can use the several levels for Oracle Database Vault trace events.
- Performance Effect of Enabling Oracle Database Vault Trace Files
  Be careful about enabling trace files.
- Enabling Oracle Database Vault Trace Events
  You can use the `ALTER SESSION` or `ALTER SYSTEM` SQL statements to enable Oracle Database Vault trace events.
- Finding Oracle Database Vault Trace File Data
  The Linux `grep` command and the ADR Command Interpreter (`ADRCI`) command-line utility can find Oracle Database Vault trace file data.
- Example: Low Level Oracle Database Vault Realm Violations in a Trace File
  You can use trace file data to track low level realm violations.

About Using Trace Files to Diagnose Oracle Database Vault Events

You can monitor the Oracle Database Vault database instance for server and background process events by enabling and checking the database instance trace files.

Trace files reveal the Oracle Database Vault policy authorization success and failures. They are useful for providing information to help resolve bug and other issues that may occur.

To set tracing for Oracle Database Vault, you must have the `DV_ADMIN` role. To perform the configuration, you use either of the `ALTER SESSION SET EVENTS` or `ALTER SYSTEM SET EVENTS` SQL statements.

> **See Also:**
>
> *Oracle Database Administrator's Guide* for more information about how to manage trace files

Types of Oracle Database Vault Trace Events That You Can and Cannot Track

You can use trace files to track a variety of Oracle Database Vault activities.

Table E-1 describes these activities.

Table E-1 Contents of Oracle Database Vault Trace Files

Database Vault Feature	Description
Realm authorizations	The trace file tracks cases of realm authorization with a rule set and realm authorization to a role. See Example: Low Level Oracle Database Vault Realm Violations in a Trace File for examples of this type of trace file.
Rule set evaluations	The trace file includes information about a rule set evaluation from a realm authorization, for a command rule, the CONNECT command rule, and from a factor.
Oracle Data Pump authorization	The trace file includes Database Vault Data Pump authorization results and other user, object, and SQL text information.

Table E-1 (Cont.) Contents of Oracle Database Vault Trace Files

Database Vault Feature	Description
Oracle Scheduler job authorization	The trace file includes the Database Vault Oracle Scheduler job authorization results, job name, job owner, current statement, and so on.
Object privilege bypass	The trace file tracks both direct grants and grants through a role. This type of trace is useful for cases where mandatory realms are not enabled, which enables users who have an object privilege to access realm protected objects.
Factor loading	The trace file tracks the expression and value for each factor loaded.
Others	Object owner bypassed realm protection and other Database Vault failed and succeeded operations

Levels of Oracle Database Vault Trace Events

You can use the several levels for Oracle Database Vault trace events.

These levels are as follows:

- **Low** prints the information for all failed Oracle Database Vault authorizations to a trace file. This type of trace file includes failed realm authorizations, failed factor loading, failed rule set evaluating, and so on. It has a low impact on Oracle Database performance.
- **High** prints trace records that include both successful and failed authorizations. Because this type of tracing tracks all the authorizations, the overhead is larger than that of the low level tracing. In addition, the trace files are usually larger.
- **Highest** prints the PL/SQL stack and function call stack to a trace file, as well as what is traced at level high (as described in Table E-1). It has the highest impact on Oracle Database performance.

Performance Effect of Enabling Oracle Database Vault Trace Files

Be careful about enabling trace files.

Doing so can increase the overhead of the database instance operation, which could decrease performance.

Enabling Oracle Database Vault Trace Events

You can use the `ALTER SESSION` or `ALTER SYSTEM` SQL statements to enable Oracle Database Vault trace events.

- Enabling Trace Events for the Current Database Session
  You can use the `ALTER SESSION SET EVENTS` SQL statement to enable trace events for the current database session.
- Enabling Trace Events for All Database Sessions
  You can use the `ALTER SYSTEM SET EVENTS` SQL statement to enable Database Vault trace events for all database sessions.

- Management of Trace Events in a Multitenant Environment
  You should be aware of how enabling trace events is affected in a multitenant environment.

Enabling Trace Events for the Current Database Session

You can use the `ALTER SESSION SET EVENTS` SQL statement to enable trace events for the current database session.

1. Log into the database instance as a user who has been granted the `DV_ADMIN` role and the `ALTER SESSION` system privilege.

   For example:

   ```
   sqlplus leo_dvowner
   Enter password: password
   Connected.
   ```

2. Enter the `ALTER SESSION SET EVENTS` SQL statement to set the tracing to low, high, or highest, as described in Levels of Oracle Database Vault Trace Events.

   - To turn on tracing for failed operations that have a low impact, enter one of the following statements:

     ```
     ALTER SESSION SET EVENTS 'TRACE[DV] DISK=LOW';

     ALTER SESSION SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 1';
     ```

   - To turn on tracing for both failed and successful operations that have a high impact, enter one of the following statements:

     ```
     ALTER SESSION SET EVENTS 'TRACE[DV] DISK=HIGH';

     ALTER SESSION SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 3';
     ```

   - To turn on tracing for both failed and successful operations with a function and PL/SQL call stack that has the highest impact, enter one of the following statements:

     ```
     ALTER SESSION SET EVENTS 'TRACE[DV] DISK=HIGHEST';

     ALTER SESSION SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 4';
     ```

Enabling Trace Events for All Database Sessions

You can use the `ALTER SYSTEM SET EVENTS` SQL statement to enable Database Vault trace events for all database sessions.

1. Log into the database instance as a user who has been granted the `DV_ADMIN` role and the `ALTER SYSTEM` system privilege.

   For example:

   ```
   sqlplus leo_dvowner
   Enter password: password
   Connected.
   ```

2. Enter the `ALTER SYSTEM SET EVENTS` SQL statement, using the syntax that is shown in Step 2 in Enabling Trace Events for the Current Database Session.

   For example:

   ```
   ALTER SYSTEM SET EVENTS 'TRACE[DV] DISK=LOW';
   ```

3. Restart the database.

   For example:

   ```
   SHUTDOWN IMMEDIATE
   STARTUP
   ```

Another way that you can enable trace events for all database sessions is to add the following line to the `init.ora` file, and then restart the database:

```
event="47998 trace name context forever, level [trace_level]"
```

Replace *`trace_level`* with one of the following values:

- `1` for the lowest level of tracing
- `3` for the high level
- `4` for the highest level

For example:

```
event="47998 trace name context forever, level [1]"
```

Management of Trace Events in a Multitenant Environment

You should be aware of how enabling trace events is affected in a multitenant environment.

- **Trace events for the current session:** In a multitenant environment, running the `ALTER SESSION SET EVENTS` SQL statement from either the root or a pluggable database (PDB) enables tracing for the current user session. If you switch from one PDB to another PDB (by using the `ALTER SESSION SET CONTAINER` statement), then tracing is still enabled for the new PDB. You cannot enable tracing for a single PDB in a multitenant container database (CDB); it applies to all PDBs and the root. Remember that must have the `ALTER SESSION SET CONTAINER` system privilege to move from one PDB to another.
- **Trace events for all database sessions:** In a multitenant environment, running the `ALTER SYSTEM SET EVENTS` statement from either the root or a specific PDB enables tracing for all PDBs in the container database.

Finding Oracle Database Vault Trace File Data

The Linux `grep` command and the ADR Command Interpreter (`ADRCI`) command-line utility can find Oracle Database Vault trace file data.

- Finding the Database Vault Trace File Directory Location
  You can find the full directory location of trace files by querying the `V$DIAG_INFO` dynamic view.
- Using the Linux grep Command to Search Trace Files for Strings
  To query or process the trace files, you can use the Linux `grep` command to search for strings.
- Using the ADR Command Interpreter (ADRCI) Utility to QueryTrace Files
  You can query trace files by using the ADR Command Interpreter (`ADRCI`) command-line utility.

Finding the Database Vault Trace File Directory Location

You can find the full directory location of trace files by querying the V$DIAG_INFO dynamic view.

- Query the V$DIAG_INFO dynamic view as follows:

```
SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Default Trace File';
```

Output similar to the following appears:

```
VALUE
--------------------------------------------------------------------------------
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_7174.trc
```

Using the Linux grep Command to Search Trace Files for Strings

To query or process the trace files, you can use the Linux grep command to search for strings.

- For example, to find the trace files that show realm authorization failures, enter the following command:

```
grep 'Result=Realm Authorization Failed'  *.trc
```

Using the ADR Command Interpreter (ADRCI) Utility to QueryTrace Files

You can query trace files by using the ADR Command Interpreter (ADRCI) command-line utility.

- To use the ADRCI utility to find trace file information, use the SHOW command.

For example, to use ADRCI to find the trace files, enter the SHOW TRACEFILE command:

```
adrci --To start ACRCI from the command line
adrci> show tracefile

diag/rdbms/orcl/orcl/trace/orcl_m002_14551.trc
diag/rdbms/orcl/orcl/trace/orcl_tmon_13450.trc
diag/rdbms/orcl/orcl/trace/orcl_vktm_963.trc
diag/rdbms/orcl/orcl/trace/alert_orcl.log
...
```

To find the number of all trace incidents:

```
adrci> show incident

ADR Home = /u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl:
*************************************************************************
234 rows fetched
```

The following ADRCI command returns a list of all trace files whose name contains the word ora:

```
adrci> show tracefile %ora%

/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_18841.trc
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_12017.trc
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_19372.trc
```

```
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_12221.trc
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_1600.trc
...
```

The following ADRCI command searches for trace files that contain the phrase Realm Authorization Failed:

```
adrci> show trace %trc -xp "[payload like '%Realm Authorization Failed%']"
```

See Also:

- *Oracle Database Utilities* for detailed information about the ADRCI utility
- *Oracle Database Administrator's Guide* for information about viewing reports with the ADRCI utility

Example: Low Level Oracle Database Vault Realm Violations in a Trace File

You can use trace file data to track low level realm violations.

Example E-1 shows an example of tracking low lever real violations.

Example E-1 Low Level Oracle Database Vault Realm Violations in a Trace File

```
*** 2010-02-05 18:35:31.438
*** SESSION ID:(34.559) 2010-02-05 18:35:31.438
*** CLIENT ID:() 2010-02-05 18:35:31.438
*** SERVICE NAME:(SYS$USERS) 2010-02-05 18:35:31.438
*** MODULE NAME:(SQL*Plus) 2010-02-05 18:35:31.438
*** ACTION NAME:() 2010-02-05 18:35:31.438

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=INSERT INTO U1.T1 VALUES(30)

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=DELETE FROM U1.T1

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T3  Object_Type=TABLE
        SQL_Text=CREATE TABLE U1.T3(C INT)

*** 2010-02-05 18:35:34.465

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
```

```
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=INSERT INTO U1.T1 VALUES(30)

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=DELETE FROM U1.T1
```

Example: High Level Trace Enabled for Oracle Database Vault Authorization

You can track Oracle Database Vault authorizations in a trace file with high level trace enabled.

Example E-2 shows an example of this type of trace file.

Example E-2 High Level Trace Enabled for Oracle Database Vault Authorization

```
Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$AUDIT  Object_Type=TABLE

Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$AUDIT  Object_Type=TABLE

Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$POL    Object_Type=TABLE

Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$USER_LOGON     Object_Type=VIEW

……

Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$POL    Object_Type=TABLE

Result=Set Factor Value
        Factor_Name=Sensitive_Treatments        Factor_Expression=/SURGERY/
PSYCHOLOGICAL

Result=Set Factor Value
        Factor_Name=Database_Instance
Factor_Expression=UPPER(SYS_CONTEXT('USERENV','INSTANCE'))      Factor_Value=1

Result=Set Factor Value
        Factor_Name=Client_IP
Factor_Expression=UPPER(SYS_CONTEXT('USERENV','IP_ADDRESS'))    Factor_Value=

Result=Set Factor Value
        Factor_Name=Authentication_Method
```

```
Factor_Expression=UPPER(SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD'))
Factor_Value=PASSWORD
……

*** ACTION NAME:() 2010-02-05 18:47:19.540

Result=Rule Set Evaluation Failed
        Command=SELECT  RuleSet_ID=2    RuleSet_Name=Disabled
        Current_User=SYSTEM
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=SELECT * FROM U1.T1

Result=Rule Set Evaluation Succeeded
        Command=SELECT  RuleSet_ID=1    RuleSet_Name=Enabled
        Current_User=SYSTEM
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=SELECT * FROM U1.T1
```

Example: Highest Level Traces on Violations on Realm-Protected Objects

You can track high level violations using trace files.

Example E-3 shows how highest level violations that involve Oracle Scheduler jobs authorization can appear in a trace file when trace is enabled at the highest level.

Example E-3 Highest Level Traces on Violations on Realm-Protected Objects

```
------ Call Stack Trace ------
kzvdvechk<-kzvdveqau<-kksfbc<-opiexe<-kpoal8<-opiodr<-ttcpip<-opitsk<-opiino<-
opiodr<-opidrv<-sou2o<-opimai_real<-ssthrdmain<-main<-__libc_start_main<-_start

Result=Object Privilege check passed
        Current_User=INVOKER2   Used_Role=1
        Object_Owner=SYSTEM     Object_Name=PRODUCT_PRIVS       Object_Type=VIEW
        SQL_Text=SELECT CHAR_VALUE FROM SYSTEM.PRODUCT_PRIVS WHERE
(UPPER('SQL*PLUS') LIKE UPPER(PRODUCT)) AND ((USER LIKE USERID) OR (USERID =
'PUBLIC')) AND (UPPER(ATTRIBUTE) = 'ROLES')
*** MODULE NAME:(SQL*Plus) 2010-02-05 18:57:53.973
*** ACTION NAME:() 2010-02-05 18:57:53.973

----- Current SQL Statement for this session (sql_id=2sr63rjm45yfh) -----
UPDATE INVOKER1.T1 SET A = 20
----- PL/SQL Stack -----
----- PL/SQL Call Stack -----
  object      line  object
  handle    number  name
0x26a00e34         1  anonymous block
0x2495b000       185  package body SYS.DBMS_ISCHED
0x24958fb8       486  package body SYS.DBMS_SCHEDULER
0x247bbb34         1  anonymous block

------ Call Stack Trace ------
kzvdvechk<-kzvdveqau<-kksfbc<-opiexe<-opipls<-opiodr<-__PGOSF151_rpidrus<-skgmstack<-
rpidru<-rpiswu2<-rpidrv<-psddr0<-psdnal<-pevm_EXECC<-pfrinstr_EXECC<-pfrrun_no_tool<-
pfrrun<-plsql_run<-peicnt<-kkxexe<-opiexe<-kpoal8<-opiodr<-kpoodr<-upirtrc<-kpurcsc<-
kpuexec
<-OCIStmtExecute<-jslvec_execcb<-jslvswu<-jslve_execute0<-jskaJobRun<-jsiRunJob<-
jsaRunJob<-spefcmpa<-spefmccallstd<-pextproc<-__PGOSF495_peftrusted<-
__PGOSF522_psdexsp<-rpiswu2<-psdextp<-pefccal<-pefcal<-pevm_FCAL<-pfrinstr_FCAL<-
```

```
pfrrun_no_tool<-pfrrun<-plsql_run
<-peicnt<-kkxexe<-opiexe<-kpoal8<-opiodr<-ttcpip<-opitsk<-opiino<-opiodr<-opidrv<-
sou2o<-opimai_real<-ssthrdmain<-main<-__libc_start_main<-_start

Result=Realm Authorization Succeeded
        Realm_Name=jobowner realm       Used_Auth_Level=0
        Current_User=119
        Object_Owner=INVOKER1   Object_Name=T1  Object_Type=TABLE
        SQL_Text=UPDATE INVOKER1.T1 SET A = 20

Result=Scheduler Job Authorization Succeeded
        Current_User=JOBOWNER   Logon_User=INVOKER2
        Job_Owner=JOBOWNER      Job_Name=DMLJOB1
        Object_Owner=INVOKER1   Object_Name=T1  Object_Type=TABLE
        SQL_Text=UPDATE INVOKER1.T1 SET A = 20
```

Disabling Oracle Database Vault Trace Events

You can disable tracing for Oracle Database Vault events.

- Disabling Trace Events for the Current Database Session
  You can use the `ALTER SESSION SET EVENTS` SQL statement to disable Database Vault tracing for the current database session.
- Disabling Trace Events for All Database Sessions
  You can use the `ALTER SYSTEM SET EVENTS` SQL statement to disable Database Vault tracing for all database sessions.
- Disabling Trace Events in a Multitenant Environment
  You should be aware of how enabling trace events is affected in a multitenant environment.

Disabling Trace Events for the Current Database Session

You can use the `ALTER SESSION SET EVENTS` SQL statement to disable Database Vault tracing for the current database session.

1. Log into the database instance as a user who has been granted the `DV_ADMIN` role and the `ALTER SESSION` system privilege.

   For example:

   ```
   sqlplus leo_dvowner
   Enter password: password
   Connected.
   ```

2. Enter both of the following SQL statements to disable tracing:

   ```
   ALTER SESSION SET EVENTS 'TRACE[DV] OFF';
   ALTER SESSION SET EVENTS '47998 trace name context off';
   ```

   Alternatively, you can use the `ALTER SYSTEM` statement as well:

   ```
   ALTER SYSTEM SET EVENTS 'TRACE[DV] OFF';
   ALTER SYSTEM SET EVENTS '47998 trace name context off';
   ```

Disabling Trace Events for All Database Sessions

You can use the `ALTER SYSTEM SET EVENTS` SQL statement to disable Database Vault tracing for all database sessions.

1. Log into the database instance as a user who has been granted the `DV_ADMIN` role and the `ALTER SYSTEM` system privilege.

   For example:

   ```
   sqlplus leo_dvowner
   Enter password: password
   Connected.
   ```

2. Enter the `ALTER SYSTEM SET EVENTS` SQL statement, using the syntax that is shown in Step 2 in Disabling Trace Events for the Current Database Session.

   For example:

   ```
   ALTER SYSTEM SET EVENTS 'TRACE[DV] OFF';
   ```

3. Restart the database.

   For example:

   ```
   SHUTDOWN IMMEDIATE
   STARTUP
   ```

Another way that you can disable trace events for all database sessions is to add the following line to the `init.ora` file, and then restart the database:

```
event="47998 trace name context off"
```

Ensure that the `init.ora` file does not have any conflicting `47998` lines, such as `event="47998 trace name context forever, level [1]"`.

Disabling Trace Events in a Multitenant Environment

You should be aware of how enabling trace events is affected in a multitenant environment.

- **Trace events for the current session:** In a multitenant environment, running the `ALTER SESSION SET EVENTS` SQL statement from either the root or a PDB disables tracing for the current user session. If you switch from one PDB to another PDB (by using the `ALTER SESSION SET CONTAINER` statement), then tracing is still disabled for the new PDB. You cannot disable tracing for a single PDB in a CDB; it applies to all PDBs and the root. Remember that must have the `ALTER SESSION SET CONTAINER` system privilege to move from one PDB to another.
- **Trace events for all database sessions:** In a multitenant environment, running the `ALTER SYSTEM SET EVENTS` statement from either the root or a specific PDB disables tracing for all PDBs in the CDB.

General Diagnostic Tips

Oracle provides general tips for diagnosing problems in realms, factors, and rule sets.

These guidelines are as follows:

- For realm protections, verify that a user has the underlying system or object privileges (granted directly or through a role) that might affect the command.
- If a realm authorization is not working, verify that the account roles are set correctly.

- For PL/SQL expressions used in factors and rule sets, grant the `EXECUTE` privilege on the PL/SQL package functions used in these expressions directly to the account and determine if the results appear to be correct.
- Use the auditing reports to diagnose problems in general. See Oracle Database Vault Auditing Reports for more information.

Configuration Problems with Oracle Database Vault Components

Oracle Database Vault provides reports to check configuration problems with realms, command rules, factors, rule sets, or secure application roles.

See the following sections for more information:

- Command Rule Configuration Issues Report
- Factor Configuration Issues Report
- Factor Without Identities Report
- Identity Configuration Issues Report
- Realm Authorization Configuration Issues Report
- Rule Set Configuration Issues Report
- Secure Application Configuration Issues Report

To run these reports, see Running the Oracle Database Vault Reports.

Resetting Oracle Database Vault Account Passwords

Backup accounts can help you reset lost passwords for users who have been granted the `DV_OWNER` and `DV_ACCTMGR` roles.

- Resetting the DV_OWNER User Password
  You can use the `DV_OWNER` backup account to reset the `DV_OWNER` user password.
- Resetting the DV_ACCTMGR User Password
  You can use the `DV_ACCTMGR` backup account to reset the `DV_ACCTMGR` user password.

Resetting the DV_OWNER User Password

You can use the `DV_OWNER` backup account to reset the `DV_OWNER` user password.

To reset the `DV_OWNER` user password, you must temporarily revoke the `DV_OWNER` role from this user, reset the password, and then re-grant the role back to the user.

1. Log in to the database instance as the backup user for the `DV_OWNER` user account.

   For example:

   ```
   sqlplus dbv_owner_backup
   Enter password: password
   ```

2. Revoke the `DV_OWNER` role from the `DV_OWNER` user who has lost the password.

   For example:

```
REVOKE DV_OWNER FROM sec_admin_owen;
```

3. Connect as a user who has been granted the DV_ACCTMGR role.

   For example:

   ```
   CONNECT accts_admin_ace
   Enter password: password
   ```

4. Reset the password for the DV_OWNER user.

   ```
   ALTER USER sec_admin_owen IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

5. Connect as the backup DV_OWNER user.

   ```
   CONNECT dbv_owner_backup
   Enter password: password
   ```

6. Grant the DV_OWNER role back to the DV_OWNER user.

   ```
   GRANT DV_OWNER TO sec_admin_owen;
   ```

> **Note:**
>
> Ensure that the backup DV_OWNER account is safely stored in case it is needed again.

Resetting the DV_ACCTMGR User Password

You can use the DV_ACCTMGR backup account to reset the DV_ACCTMGR user password.

To reset the DV_ACCTMGR user password, you can use the backup DV_ACCTMGR account to reset this user's password.

1. Log in to the database instance as the backup user for the DV_ACCTMGR user account.

   For example:

   ```
   sqlplus dbv_acctmgr_backup
   Enter password: password
   ```

2. Reset the password for the DV_ACCTMGR user.

   For example:

   ```
   ALTER USER accts_admin_ace IDENTIFIED BY password;
   ```

   Follow the guidelines in *Oracle Database Security Guide* to replace *password* with a password that is secure.

> **Note:**
>
> Ensure that the backup `DV_ACCTMGR` account is safely stored in case it is needed again.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X