Oracle® Database Vault

Administrator's Guide

12*c* Release 1 (12.1)

E49109-08

December 2014

Oracle Database Vault Administrator's Guide 12*c* Release 1 (12.1)

E49109-08

Primary Author: Patricia Huey

Contributor: The Oracle Database 12c documentation is dedicated to Mark Townsend, who was an inspiration to all who worked on this release.

Contributors: Tammy Badnar, Tom Best, Todd Bottger, Ji-won Byun, Ben Chang, Martin Cheng, Chi Ching Chui, Scott Gaetjen, Viksit Gaur, Lijie Heng, Dominique Jeunot, Peter Knaggs, Chon Lee, Paul Needham, Yi Ouyang, Hozefa Palitanawala, Robert Pang, Gayathri Sairamkrishnan, Vipin Samar, James Spiller, Srividya Tata, Kamal Tbeileh, Saravana Soundararajan, Sudheesh Varma, Peter Wahl, Rodney Ward

Contents

List of Examples

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.

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

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

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/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 12c Release 1 (12.1.0.2)
- Changes in Oracle Database Vault 12c Release 1 (12.1.0.1)

Changes in Oracle Database Vault 12c Release 1 (12.1.0.2)

The following are changes in *Oracle Database Vault Administrator's Guide* for Oracle Database 12*c* Release 1 (12.1.0.2):

- New Features

New Features

Topics:

- Additional Privileges for the DV_ACCTMGR Role
- Support for the Oracle Label Security Function OLS_LABEL_DOMINATES in Rules
- Improvements for the Profile Used to Store Privileges for Privilege Analysis

Additional Privileges for the DV_ACCTMGR Role

In this release, users who have been granted the `DV_ACCTMGR` role can manage users who have been granted the `DV_MONITOR`, `DV_SECANALYST`, and `DV_AUDIT_CLEANUP` roles. This functionality enables you to easily change the password for user accounts such as the `DBSNMP` user, which often expires, and which have roles such as `DV_MONITOR`.

See the following sections for more information about these roles:

- "DV_MONITOR Database Vault Monitoring Role" on page 12-9
- "DV_SECANALYST Database Vault Security Analyst Role" on page 12-10
- "DV_AUDIT_CLEANUP Audit Trail Cleanup Role" on page 12-10

Support for the Oracle Label Security Function OLS_LABEL_DOMINATES in Rules

When creating rules, you now can include the public standalone function `OLS_LABEL_DOMINATES`, which is also new to this release.

See the following sections for more information:

- "CREATE_RULE Procedure" on page 14-3 for an example of creating a rule that uses the `OLS_LABEL_DOMINATES` function
- *Oracle Label Security Administrator's Guide* for more information about the `OLS_LABEL_DOMINATES` function

Improvements for the Profile Used to Store Privileges for Privilege Analysis

Starting with this release, the captured privileges for pre-compiled database objects such as PL/SQL packages are stored in the `ORA$DEPENDENCY` profile. This profile cannot be deleted.

See "How Privilege Analysis Works with Pre-Compiled Database Objects" on page 4-2 for more information about how privilege analysis works with precompiled database objects.

Changes in Oracle Database Vault 12*c* Release 1 (12.1.0.1)

The following are changes in *Oracle Database Vault Administrator's Guide* for Oracle Database 12*c* Release 1 (12.1.0.1):

- New Features
- Deprecated Features
- Desupported Feature

New Features

The following features are new in this release:

- Changes to Registering Oracle Database Vault
- Changes to Enabling or Disabling Oracle Database Vault
- Simplified Oracle Database Vault Administration
- Default Protection for Database Vault Schemas
- Performing Privilege Analysis for System and Object Privileges
- New Oracle Database Vault Realms
- Changed Oracle Database Vault Realms
- Mandatory Realm Protection
- Additional Audit Information for Realms
- Additional DBMS_MACADM PL/SQL Authorization Procedures
- Changes to Oracle Data Pump and Oracle Scheduler in Oracle Database Vault
- Integration in a Multitenant Environment
- Ability to Control the Use of the ORADEBUG Utility
- Inclusion of Database Vault Records in the Unified Audit Trail
- Auditing of the Oracle Database Vault Administrator's Actions
- New Data Dictionary Views to Capture Audit Data
- New Role for Cleaning the Oracle Database Vault Audit Trail

Changes to Registering Oracle Database Vault

In previous releases of Oracle Database Vault, you registered (that is, enabled) Database Vault with a database by using Oracle Database Configuration Assistant (DBCA). You now can register Database Vault from the command line, by using the DVSYS.CONFIGURE_DV and DBMS_MACADM.ENABLE_DV procedures.

See the following sections for more information:

- "Registering Oracle Database Vault with an Oracle Database" on page 3-1
- "DVSYS.CONFIGURE_DV General System Maintenance Procedure" on page 20-13
- "ENABLE_DV Procedure" on page 20-11

Changes to Enabling or Disabling Oracle Database Vault

This release introduces two new DBMS_MACADM procedures that you can use when you enable or disable Oracle Database Vault: DBMS_MACADM.ENABLE_DV and DBMS_MACADM.DISABLE_DV.

See the following sections for more information:

- Appendix B, "Disabling and Enabling Oracle Database Vault"
- "DISABLE_DV Procedure" on page 20-10
- "ENABLE_DV Procedure" on page 20-11

Simplified Oracle Database Vault Administration

Starting with this release, you can access Database Vault Administrator (DVA) from Oracle Enterprise Manager. DVA is no longer an independent application. The advantage of this design is that it makes it easier for you to create, view, and modify Oracle Database Vault settings from a centralized location when you are working with Oracle Database.

Another enhancement is that all of the configuration-related DBMS_MACADM PL/SQL package procedures and functions have been incorporated into the Enterprise Manager interface. For example, you now can delete rules from a rule set using the interface.

See "Logging into Oracle Database Vault" on page 3-7 for more information.

Default Protection for Database Vault Schemas

Because the Database Vault metadata (for example, policy and factors) as well as the administrative packages reside in the DVSYS and DVF schemas, these schemas are critical to the integrity of Oracle Database Vault. Starting from this release, direct login to these accounts will be blocked by default. You can disable or re-enable this default protection by using two new DBMS_MACADM procedures: DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS and DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS. You must have the DV_OWNER role to execute these procedures.

Performing Privilege Analysis for System and Object Privileges

You now can easily identify excessive system and object privilege grants or roles and privileges that are not being used, even with large numbers of user accounts in complex Oracle Database installations. A privilege analysis finds privilege usage according to a specified condition. For example, the privilege analysis can record privileges that are used to run a given application module and privileges that are used by a given logon user. The results of the privilege analysis are stored in data dictionary views.

See Chapter 4, "Performing Privilege Analysis to Find Privilege Use," for more information.

New Oracle Database Vault Realms

- **Oracle Default Schema Protection realm.** This realm protects roles and schemas that relate to Oracle Database options such as Oracle OLAP, Oracle Spatial, and Oracle Text.
- **Oracle System Privilege and Role Management realm.** This realm protects sensitive roles that are used for exporting and importing data to and from a database. It also contains user authorizations for users who must be able to grant system privileges.
- **Oracle Default Component Protection realm.** This realm protects the `SYSTEM` and `OUTLN` schemas.

Changed Oracle Database Vault Realms

- **Oracle Database Vault realm.** This realm protects configuration and role information in the `DVSYS`, `DVF`, and `LBACSYS` schemas. Starting with this release, the `SYS.AQ%DATAPUMP%` tables and view have been removed from this realm. These objects no longer need realm protection because Oracle Streams Advanced Queuing and Oracle Data Pump authorizations are now handled differently.

  See the following sections for more information:

  – "Privileges for Using Oracle Streams with Oracle Database Vault" on page 11-14

  – "Using Oracle Data Pump with Oracle Database Vault" on page 11-4
- **Database Vault Account Management realm.** This realm is designed for administrators who are responsible for creating and managing database accounts and database profiles. Starting with this release, the account manager (role `DV_ACCTMGR`) can grant the `CREATE SESSION` privilege to users.

Mandatory Realm Protection

In previous releases of Oracle Database Vault, object owners or users who had object privileges for objects within a realm still had access to these objects, even if these users were not realm participants or realm owners. In this release, you optionally can restrict these users' access to the objects by using a mandatory realm. In a mandatory realm, only realm participants and realm owners can access realm-protected objects.

Mandatory realms enable you to have more flexibility when you create Database Vault policies and they help you to achieve more security. Additional advantages of mandatory realms are that in some cases they prevent roles from being altered to achieve access based on role authorization, and they are useful during patch upgrades. Roles that are protected by a mandatory realm cannot be modified. Therefore, no privileges can be granted to or revoked from a role once it is protected by a mandatory realm.

See "Using Mandatory Realms to Restrict User Access to Objects within a Realm" on page 5-2 for more information.

Additional Audit Information for Realms

In previous releases, the audit trail recorded only the last realm evaluated for any realm enforcement policies. Starting with this release, the `ACTION_OBJECT_ID` and `ACTION_OBJECT_NAME` fields of the new `DVSYS.DV$CONFIGURATION_AUDIT` and `DVSYS.DV$ENFORCEMENT_AUDIT` data dictionary views will capture all realm IDs and realm names that have the audit option set to audit on failure or audit on success or failure.

See the following sections for more information:

- "DVSYS.DV$CONFIGURATION_AUDIT View" on page 22-21
- "DVSYS.DV$ENFORCEMENT_AUDIT View" on page 22-25

Additional DBMS_MACADM PL/SQL Authorization Procedures

In this release, the following procedures have been added to the `DBMS_MACADM` PL/SQL package, to give you greater control over more specific user authorizations:

- `DBMS_MACADM.AUTHORIZE_DDL` grants users authorization to execute data definition language (DDL) statements on the specified schema. To revoke this authorization, you can use the `DBMS_MACADM.UNAUTHORIZE_DDL` procedure.
- `DBMS_MACADM.AUTHORIZE_PROXY_USER` grants users authorization to proxy other user account. To revoke this authorization, you can use the `DBMS_MACADM.UNAUTHORIZE_PROXY_USER` procedure.

See the following sections for more information:

- "AUTHORIZE_DDL" on page 20-4
- "UNAUTHORIZE_DDL Procedure" on page 20-7
- "AUTHORIZE_PROXY_USER" on page 20-4
- "UNAUTHORIZE_PROXY_USER Procedure" on page 20-8

Changes to Oracle Data Pump and Oracle Scheduler in Oracle Database Vault

In this release, the following default rule sets for Oracle Data Pump and Oracle Scheduler have been removed:

- Allow Oracle Data Pump Operation
- Allow Scheduler Job

The following data dictionary views have been added, which record information about the Oracle Data Pump and Oracle Scheduler authorizations in Database Vault:

- `DVSYS.DBA_DV_DATAPUMP_AUTH`
- `DVSYS.DBA_DV_JOB_AUTH`

The following `DBMS_MACADM` PL/SQL package procedures have been added so that you can authorize or unauthorize users to perform Oracle Data Pump transportable tablespace operations in an Oracle Database Vault environment:

- "AUTHORIZE_TTS_USER Procedure" on page 20-6
- "UNAUTHORIZE_TTS_USER Procedure" on page 20-9

See the following sections for more information:

- "Using Oracle Data Pump with Oracle Database Vault" on page 11-4
- "DVSYS.DBA_DV_DATAPUMP_AUTH View" on page 22-4
- "Using Oracle Scheduler with Oracle Database Vault" on page 11-12
- "DVSYS.DBA_DV_JOB_AUTH View" on page 22-10

Integration in a Multitenant Environment

Oracle Database 12c Release 1 (12.1) now enables you to consolidate multiple databases, called pluggable databases (PDBs), into one large Oracle database called a multitenant container database (CDB). This enables you to better manage many applications that are built on one or more Oracle databases. Databases that are protected by Oracle Database Vault can accommodate CDBs.

See the following for more information:

- "How Oracle Database Vault Works in a Multitenant Environment" on page 1-9
- *Oracle Database Administrator's Guide* for details about managing a multitenant environment

Ability to Control the Use of the ORADEBUG Utility

You can enable or disable the use of the `ORADEBUG` utility in an Oracle Database Vault environment. This enhancement enables you to prevent the misuse of critical `ORADEBUG` features such as manipulating and dumping internal structures, enabling or disabling SQL tracing for other users, and suspending intensive processes in a Database Vault-enabled database.

See "Using the ORADEBUG Utility with Oracle Database Vault" on page 11-20 for more information.

Inclusion of Database Vault Records in the Unified Audit Trail

This release introduces the unified audit trail, in which the various Oracle Database audit trails (such as the standard audit trail, fine grained audit trail, and so on, in addition to the Database Vault and Oracle Label Security audit trails) are consolidated into one audit trail. In a new installation, unified auditing is enabled. If you are upgrading from an earlier release, then you can choose between a unified or non-unified auditing environment. If you choose to use unified auditing, then Oracle Database Vault audit records are stored in a table in the Oracle Database `AUDSYS` schema, not the Database Vault `AUDIT_TRAIL$` table in the `SYS` schema.

By having the various audit trails in one location, you have one centralized view of all auditing that has occurred from different areas of the database. The unified audit records are presented in a consistent, uniform format. This makes it easier for you to run analysis reports, for example.

See *Oracle Database Security Guide* for more information about how to capture unified audit records for Oracle Database Vault.

Auditing of the Oracle Database Vault Administrator's Actions

Starting with this release, Oracle Database Vault can audit all actions performed by the Database Vault administrator. All configuration changes made to Database Vault components, such as realms or factors, are now written to the audit trail. This auditing captures the creation, modification, and deletion of most Database Vault enforcements as well as Database Vault authorization for components such as Oracle Data Pump and Oracle Scheduler. However, it does not audit by default the grants and revocations of Database Vault-protected roles, such as `DV_OWNER` and `DV_ADMIN`. This type of audit relies on the audit configuration of the relevant Oracle Database Vault realms.

See Appendix A, "Auditing Oracle Database Vault," for information about auditing.

New Data Dictionary Views to Capture Audit Data

The following unified auditing-specific data dictionary views are new for this release:

- DVSYS.DV$CONFIGURATION_AUDIT provides information about Database Vault configuration changes made by the Database Vault administrator. See "DVSYS.DV$CONFIGURATION_AUDIT View" on page 22-21 for more information.
- DVSYS.DV$ENFORCEMENT_AUDIT provides information about user activities that affect Database Vault enforcement policies. See "DVSYS.DV$ENFORCEMENT_AUDIT View" on page 22-25 for more information.

New Role for Cleaning the Oracle Database Vault Audit Trail

For better separation of duty, this release introduces the DV_AUDIT_CLEANUP role, which can be granted to any user who is responsible for cleaning the Database Vault audit trail. This role does not apply to unified auditing environments.

See the following sections for more information about using this role:

- "DV_AUDIT_CLEANUP Audit Trail Cleanup Role" on page 12-10
- "Archiving and Purging the Oracle Database Vault Audit Trail" on page A-5

Deprecated Features

The following features are deprecated in this release, and may be desupported in a future release:

- Deprecated realm:
  - **Oracle Data Dictionary realm:** In previous releases, the Oracle Data Dictionary realm protected many objects, all of which were considered highly sensitive. However, this large number of objects has made it difficult to achieve fine-grained authorization in order to meet better separation of duty standards. Starting with this release, the Oracle Data Dictionary realm has been deprecated. Instead, the objects formerly protected by the Oracle Data Dictionary realm have been migrated to the new realms that are described in this section.

  See "Default Realms" on page 5-4 for information about alternatives.
- Deprecated default rule sets:
  - **Allow Oracle Data Pump Operation rule set:** After you authorize a user to perform Data Pump operations in a Database Vault environment, you can check the user's authorization by querying the DVSYS.DBA_DV_DATAPUMP_AUTH data dictionary view, which is new for this release.
  - **Allow Scheduler Job rule set:** After you authorize a user to perform Oracle Scheduler operations, you can check this user's authorization by querying the DVSYS.DBA_DV_JOB_AUTH view, also new for this release.

  See "Default Rule Sets" on page 6-2 for information about default rule sets.
- The DBMS_MACADM.SYNC_RULES procedure has been deprecated because its functionality has been built into the rule creation functionality.
- Database Vault Administrator (DVA) has been deprecated. Its functionality is now part of the of Oracle Enterprise Manager Cloud Control interface.

  See "Logging into Oracle Database Vault" on page 3-7 for more information.

Desupported Feature

Database Vault Configuration Assistant (DVCA) is desupported starting with this release. The functionality for DVCA has been replaced with PL/SQL equivalents.

See the following sections for more information:

- Appendix B, "Disabling and Enabling Oracle Database Vault"
- Appendix C, "Postinstallation Oracle Database Vault Procedures"

1 Introducing Oracle Database Vault

Oracle Database Vault enables you to control administrative access to your data. You should be aware of the privileges necessary to use Database Vault, the components of Database Vault, and how Database Vault works in various scenarios.

Topics:

- What Is Oracle Database Vault?
- What Privileges Do You Need to Use Oracle Database Vault?
- Components of Oracle Database Vault
- How Oracle Database Vault Addresses Compliance Regulations
- How Oracle Database Vault Protects Privileged User Accounts
- How Oracle Database Vault Allows for Flexible Security Policies
- How Oracle Database Vault Addresses Database Consolidation Concerns
- How Oracle Database Vault Works in a Multitenant Environment

What Is Oracle Database Vault?

Oracle Database Vault provides controls for privileged accounts and database configurations. You can use Database Vault in Enterprise Manager. In addition, you can perform privilege analysis.

Topics:

- About Oracle Database Vault
- Controls for Privileged Accounts
- Controls for Database Configuration
- Enterprise Applications Protection Policies
- Run-time Privilege Analysis for Users and Applications

About Oracle Database Vault

Oracle Database Vault provides powerful security controls to help protect application data from unauthorized access, and comply with privacy and regulatory requirements.

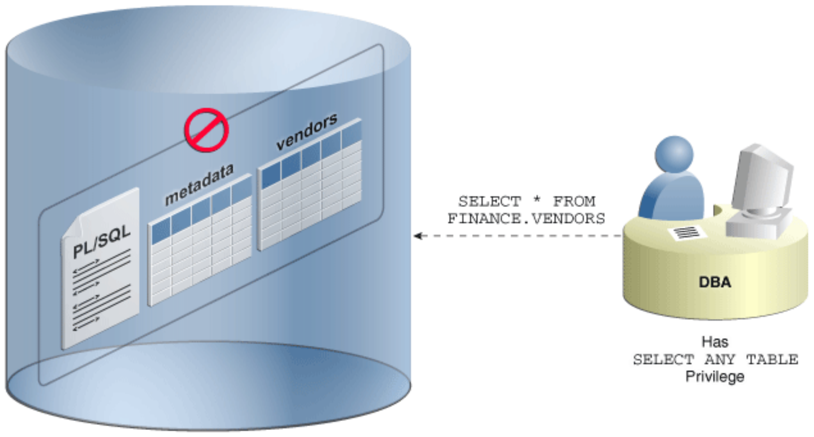
You can deploy controls to block privileged account access to application data and control sensitive operations inside the database using multi-factor authorization. Through the analysis of privileges and roles, you can increase the security of existing applications. Oracle Database Vault secures existing 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

Among the more common audit findings are unauthorized changes to database entitlements, including grants of the `DBA` role, as well as new accounts and database objects.

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 multi-factor 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 mandatory realms enable you to seal off access to application objects, even to those with direct object grants, including the object owner. This is useful when Support must access the application schema directly as the application owner. 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 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 enable you to perform privilege analysis, which helps increase the security of applications by identifying the actual privileges and roles 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 a set of 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 is created during the process, and the `DV_ACCTMGR` role is granted to a second, optional user. You can grant the Database Vault roles to other users, but ensure that these users are trusted.

See Also: "Oracle Database Vault Roles" on page 12-2 for a full list of Oracle Database Vault roles

Components of Oracle Database Vault

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

Topics:

- Oracle Database Vault Access Control Components
- Oracle Enterprise Manager Cloud Control Database Vault Administrator Pages
- Oracle Database Vault DVSYS and DVF Schemas
- Oracle Database Vault PL/SQL Interfaces and Packages
- Oracle Database Vault Reporting and Monitoring 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.

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. Chapter 5, "Configuring Realms," discusses realms in detail. See also Chapter 13, "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. Chapter 7, "Configuring Command Rules," discusses command rules in detail. See also Chapter 15, "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. 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. Chapter 8, "Configuring Factors," discusses factors in detail. See also Chapter 16, "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. Chapter 6, "Configuring Rule Sets," discusses rule sets in detail. See also Chapter 14, "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. Chapter 9, "Configuring Secure Application Roles for Oracle Database Vault," discusses secure application roles in detail. See also Chapter 17, "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" on page 1-5 provides an overview.

In addition to these components, you can use the `DBMS_PRIVILEGE_CAPTURE` PL/SQL package and related data dictionary views to analyze the privilege use of your users. Based on this information, you can build Oracle Database Vault policies to monitor this privilege use. Chapter 4, "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. Chapter 3, "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. Chapter 24, "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.

Chapter 3 through Chapter 11 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. Chapter 23 explains how to use these pages to monitor Database Vault activity, and Chapter 24, "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.

Chapter 12, "Oracle Database Vault Schemas, Roles, and Accounts," describes these schemas in detail.

Oracle Database Vault PL/SQL Interfaces and Packages

Oracle Database Vault provides a collection of PL/SQL interfaces and packages that enable security managers or application developers to configure the access control policy as required.

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 Chapter 13 through Chapter 21 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.

See Chapter 24, "Oracle Database Vault Reports," for more information about the reports that you can generate. Chapter 23, "Monitoring Oracle Database Vault," explains how to monitor Oracle Database Vault.

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:

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

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

How Oracle Database Vault Protects Privileged User Accounts

Most recent security breaches have revealed that intruders, whether coming from inside or from outside an organization, are increasingly targeting privileged users database accounts to compromise them and use them 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, such as `SYSTEM`, 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 potentially expose large amounts of 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.

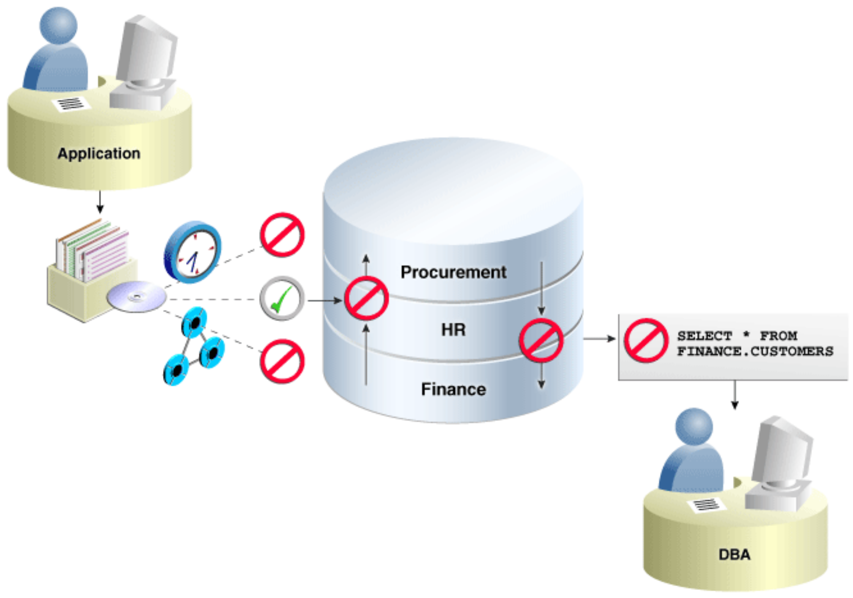
Oracle Database Vault provides three distinct separation of duty controls out-of-the-box for security administration, account management, and day-to-day database administration activities. Oracle Database Vault separation of duty controls can be customized and organizations with limited resources can assign multiple Oracle Database Vault responsibilities to the same administrator.

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

Oracle Database Vault can be used with Oracle Multitenant to provide increased security for consolidation.

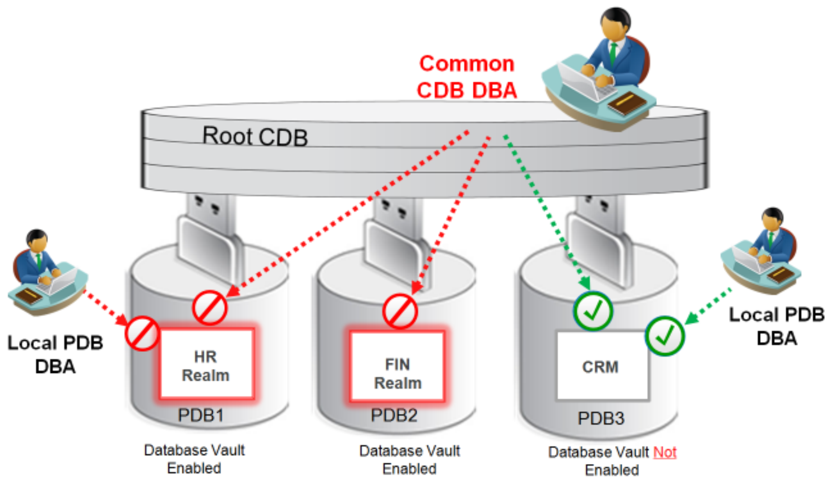
It can prevent privileged user access inside a pluggable database (PDB) and between the PDB and the common privileged user at the container database. Each pluggable database (PDB) has its own Database Vault metadata, such as realms, rule sets, command rules, default policies (such as default realms)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

Because Oracle Database Vault policies are scoped to individual PDBs, you can create individual 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.

Figure 1–3 illustrates how Database Vault protection applies to both common and local users. 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 CRM application in PDB3, where Database Vault is not enabled.

Figure 1–3   Oracle Database Vault in a Multitenant Environment



See Also:

- "Using Command Rules in a Multitenant Environment" on page 7-2
- "Plugging a Database Vault-Enabled PDB to a CDB" on page 11-18
- *Oracle Database Administrator's Guide* for detailed information about CDB

2 What to Expect After You Enable Oracle Database Vault

When you enable Oracle Database Vault affects a number of Oracle Database features, including initialization parameters, user authorizations, roles, privileges, and `AUDIT` statement settings.

Topics:

- Initialization and Password Parameter Settings That Change
- How Oracle Database Vault Restricts User Authorizations
- Using New Database Roles to Enforce Separation of Duties
- Privileges That Are Revoked from Existing Users and Roles
- Privileges That Are Prevented for Existing Users and Roles
- Modified AUDIT Statement Settings for a Non-Unified Audit Environment

> **See Also:** Appendix D, "Oracle Database Vault Security Guidelines," for guidelines on managing security in the Oracle Database configuration

Initialization and Password Parameter Settings That Change

When you install Oracle Database Vault, the installation process 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, located in `$ORACLE_HOME/srvm/admin`. For more information about this file, see *Oracle Database Administrator's Guide*.

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 SQL Language Reference*.

Table 2–1 (Cont.) Modified Database Initialization Parameter Settings

Parameter	Default Value in Database	New Value Set by Database Vault	Impact of the Change
`RECYCLEBIN`	`ON`	`OFF`	Controls whether the Flashback Drop feature is turned on or off. If `RECYCLEBIN` is set to `OFF`, then dropped tables do not go into the recycle bin. If it is set to `ON`, then dropped tables go into the recycle bin, which means that they can be recovered. **See Also:** ■ "Security Considerations for the Recycle Bin" on page D-11 ■ *Oracle Database Reference* for more information about `RECYCLEBIN`
`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`	`FALSE`	`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 SQL Language Reference*.

How Oracle Database Vault Restricts User Authorizations

During installation of Oracle Database Vault, the installer prompts for two additional 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.

For guidelines on managing separation of duty, see "Separation of Duty Guidelines" on page D-1.

Using New Database Roles to Enforce Separation of Duties

To meet regulatory, privacy and other compliance requirements, Oracle Database Vault implements the concept of *separation of duty*.

Oracle Database Vault makes clear separation between the account management responsibility, data security responsibility, and database resource management responsibility inside the database. This means that the concept of a superprivileged 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 the `SYS` user and other accounts with the `DBA` role and other 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.

See Also:

- "Separation of Duty Guidelines" on page D-1 for advice on managing separation of duty for your site
- "Oracle Database Vault Roles" on page 12-2 for detailed information about the roles created during the Oracle Database Vault installation
- "Oracle Database Vault Accounts" on page 12-19 for default accounts that are created and for suggestions of additional accounts that you may want to create

Privileges That Are Revoked from Existing Users and Roles

When you install Oracle Database Vault, it revokes a set of privileges from several Oracle Database-supplied users and roles, as part of the separation of duty enhancement.

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.

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" on page 11-4.

[2] To authorize users to schedule database jobs, see "Using Oracle Scheduler with Oracle Database Vault" on page 11-12.

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.

See Also:

- Table 12–1, " Privileges of Oracle Database Vault Roles" on page 12-5
- "DV_ACCTMGR Database Vault Account Manager Role" on page 12-16

Privileges That Are Prevented for Existing Users and Roles

Several privileges are prevented for all users and roles who have been granted these privileges, including users `SYS` and `SYSTEM`.

- `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, Database Vault configures several `AUDIT` statements in the database.

See "Oracle Database Audit Settings Created for Oracle Database Vault" on page A-8 for more information.

3 Getting Started with Oracle Database Vault

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

Topics:

- Registering Oracle Database Vault with an Oracle Database
- Logging into Oracle Database Vault
- Quick Start Tutorial: Securing a Schema from DBA Access

Registering Oracle Database Vault with an Oracle Database

You must register Oracle Database Vault before you can use it. You can register it in a non-multitenant environment and for a multitenant environment, perform the registration based on the type of configuration you have.

Topics:

- About Registering Oracle Database Vault with an Oracle Database
- Registering Oracle Database Vault in a Non-Multitenant Environment
- Registering Oracle Database Vault with a Common User to Manage the CDB Root
- Registering Database Vault Common Users to Manage Specific PDBs
- Registering Oracle Database Vault Common Users to Manage All Database Vault PDBs
- Plugging in a Database That Has Oracle Database Vault Enabled
- Verifying That Oracle Database Vault Is Configured and Enabled

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. As part of the registration process, you create the Database Vault administrative accounts. The registration process enables Oracle Label Security if it is not already enabled. This procedure applies to the current pluggable database (PDB), as well as to both single-instance and Oracle RAC installations. To check if Database Vault has already been enabled, see "Checking if Oracle Database Vault Is Enabled or Disabled" on page B-2.

This section explains how to register Oracle Database Vault in a non-multitenant environment, and several ways that you can register it in a multitenant environment.

Registering Oracle 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 user `SYS` with the `SYSDBA` administrative privilege.

   ```
   sqlplus sys as sysdba
   Enter password: password
   ```

2. As user `SYS` with the `SYSDBA` administrative privilege, configure the Database Vault user accounts.

   ```
   BEGIN
    DVSYS.CONFIGURE_DV (
      dvowner_uname         => 'dbv_owner',
      dvacctmgr_uname       => 'dbv_acctmgr');
    END;
   /
   ```

   Do not enter the names `DV_OWNER`, `DV_ACCTMGR`, or the names of any other Database Vault roles for these user accounts. These names are reserved words. "Oracle Database Vault Roles" on page 12-2 describes the Database Vault roles.

3. 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.

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

   For example:

   ```
   CONNECT dbv_owner
   Enter password: password
   ```

5. Enable Oracle Database Vault.

   ```
   EXEC DBMS_MACADM.ENABLE_DV;
   ```

6. Connect with the `SYSDBA` administrative privilege.

   ```
   CONNECT / AS SYSDBA
   ```

7. Restart the database.

   ```
   SHUTDOWN IMMEDIATE
   STARTUP
   ```

See "Verifying That Oracle Database Vault Is Configured and Enabled" on page 3-7 to confirm that this procedure was successful. After you have registered Oracle Database Vault with an Oracle database, you can start Oracle Database Vault Administrator. See "Logging into Oracle Database Vault" on page 3-7 for more information.

Registering Oracle Database Vault with a Common User to Manage the CDB Root

In SQL*Plus, 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 two user accounts to be used as the Database Vault accounts and grant them the `CREATE SESSION` and `SET CONTAINER` privileges.

   One account will be the Database Vault Owner user and the second (optional) account will be the Database Vault Account Manager account. Ensure that they have at minimum the `CREATE SESSION` privilege. If you decide not to create a Database Vault Account Manager account, then the Database Vault Owner will have the Database Vault Account Manager privileges. For better separation of duty, Oracle recommends that you create a Database Vault Account Manager account.

   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_acctmgr_root IDENTIFIED BY
   password CONTAINER = ALL;
   ```

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

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

4. Configure the Database Vault user accounts.

   ```
   BEGIN
    DVSYS.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 Database Vault Owner user that you just configured.

   For example:

   ```
   CONNECT c##dbv_owner_root
   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
   ```

See "Verifying That Oracle Database Vault Is Configured and Enabled" on page 3-7 to confirm that this procedure was successful. After you have registered Oracle Database Vault with an Oracle database, you can start Oracle Database Vault Administrator. See "Logging into Oracle Database Vault" on page 3-7 for more information.

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 two user accounts to be used as the Database Vault accounts.

   See Step 2 under "Registering Oracle Database Vault with a Common User to Manage the CDB Root" on page 3-3 for more information about these accounts.

3. 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.

4. Grant the `CREATE SESSION` and `SET CONTAINER` privileges to the users.

   For example:

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

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

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

6. While still in the PDB, configure the Database Vault user accounts.

   For example:

   ```
   BEGIN
    DVSYS.CONFIGURE_DV (
      dvowner_uname         => 'c##dbv_owner_root',
   ```

```
    dvacctmgr_uname        => 'c##dbv_acctmgr_root');
  END;
 /
```

7. 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.

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

   For example:

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

9. Enable Oracle Database Vault.

   ```
   EXEC DBMS_MACADM.ENABLE_DV;
   ```

10. Connect with the `SYSDBA` administrative privilege.

    ```
    CONNECT / AS SYSDBA
    ```

11. Close and reopen the PDB.

    For example:

    ```
    ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;

    ALTER PLUGGABLE DATABASE pdb_name OPEN;
    ```

See "Verifying That Oracle Database Vault Is Configured and Enabled" on page 3-7 to confirm that this procedure was successful. After you have registered Oracle Database Vault with an Oracle database, you can start Oracle Database Vault Administrator. See "Logging into Oracle Database Vault" on page 3-7 for more information.

Registering Oracle Database Vault Common Users to Manage All Database Vault PDBs

Using SQL*Plus, you can register Oracle Database Vault common users to manage all Database Vault PDBs.

1. In a multitenant environment, log into the root of the database instance as user `SYS` with the `SYSDBA` administrative privilege.

   ```
   sqlplus sys as sysdba
   Enter password: password
   ```

2. Ensure that Oracle Database Vault has been configured and enabled on the root.

   See "Verifying That Oracle Database Vault Is Configured and Enabled" on page 3-7.

3. If you have not already done so, create two user accounts to be used as the Database Vault accounts, and grant them the `CREATE SESSION` and `SET CONTAINER` privileges.

   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_acctmgr_root IDENTIFIED BY
   ```

```
password CONTAINER = ALL;
```

See Step 2 under "Registering Oracle Database Vault with a Common User to Manage the CDB Root" on page 3-3 for more information about these accounts.

4. As user SYS with the SYSDBA privilege, connect to the PDB to which the common users must have access.

   For example:

   ```
   CONNECT SYS@pdb_name AS SYSDBA
   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. Configure the Database Vault user accounts for the PDB.

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

6. 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.

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

   For example:

   ```
   CONNECT c##dbv_owner_root
   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.

8. Enable Oracle Database Vault.

   ```
   EXEC DBMS_MACADM.ENABLE_DV;
   ```

9. Connect with the SYSDBA administrative privilege.

   ```
   CONNECT / AS SYSDBA
   ```

10. Close and reopen the PDB.

    For example:

    ```
    ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;

    ALTER PLUGGABLE DATABASE pdb_name OPEN;
    ```

11. Repeat Step 4 through Step 10 for each PDB.

See "Verifying That Oracle Database Vault Is Configured and Enabled" on page 3-7 to confirm that this procedure was successful. After you have registered Oracle Database Vault with an Oracle database, you can start Oracle Database Vault Administrator. See "Logging into Oracle Database Vault" on page 3-7 for more information.

Plugging in a Database That Has Oracle Database Vault Enabled

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.

To enable a common user to manage the Database Vault configuration for this PDB:

1. Log into the PDB as the local Database Vault owner.

   For example:

   ```
   sqlplus dbv_owner@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.

2. Grant the common user the `CREATE SESSION` and `SET CONTAINER` privileges, and the appropriate Database Vault roles.

   For example:

   ```
   GRANT CREATE SESSION, SET CONTAINER, DV_OWNER TO c##dbv_owner_root;
   GRANT CREATE SESSION, SET CONTAINER, DV_ACCTMGR TO c##dbv_acctmgr_root;
   ```

If necessary, see "Verifying That Oracle Database Vault Is Configured and Enabled" on page 3-7. After you have registered Oracle Database Vault with an Oracle database, you can start Oracle Database Vault Administrator. See "Logging into Oracle Database Vault" on page 3-7 for more information.

Verifying That Oracle Database Vault Is Configured and Enabled

You can query the `V$OPTION` dynamic view to verify if Oracle Database is configured and enabled.

- Query the `V$OPTION` dynamic view as follows:
  - To find if Oracle Database Vault is configured and enabled, run the following query, which should show the `VALUE` setting as `TRUE`:

    ```
    SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';
    ```

  - To check if Oracle Label Security is enabled, query `V$OPTION` as follows:

    ```
    SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Label Security';
    ```

At this stage, you have only the two Oracle Database Vault accounts: one for the Database Vault Owner and the other for the Database Vault Account Manager. Oracle recommends that you reserve these accounts as back-up accounts, and then grant the `DV_OWNER` and `DV_ACCTMGR` roles to existing user accounts for every day use. See the tip under "Oracle Database Vault Accounts" on page 12-19 for more information.

Logging into Oracle Database Vault

From Oracle Enterprise Manager Cloud Control (Cloud Control), you can use the Oracle Database Vault pages to administer and monitor Database Vault-protected databases from a centralized console, automate alerts, view Database Vault reports, and propagate Database Vault policies to other Database Vault-protected databases.

1. Ensure that you have configured the Cloud Control target databases that you plan to use with Database Vault.

See the Oracle Enterprise Manager online help and *Oracle Enterprise Manager Advanced Configuration* for more information about configuring target databases.

2. If necessary, register Oracle Database Vault.

   If you have just installed Oracle Database Vault, you must register it with the database. See "Registering Oracle Database Vault with an Oracle Database" on page 3-1 for more information.

3. Start Cloud Control.

   For example:

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

4. Log into Cloud Control as user `SYSMAN`.
5. In the Cloud Control home page, from the **Targets** menu, select **Databases**.
6. 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.

7. From the **Security** menu, select **Database Vault**.

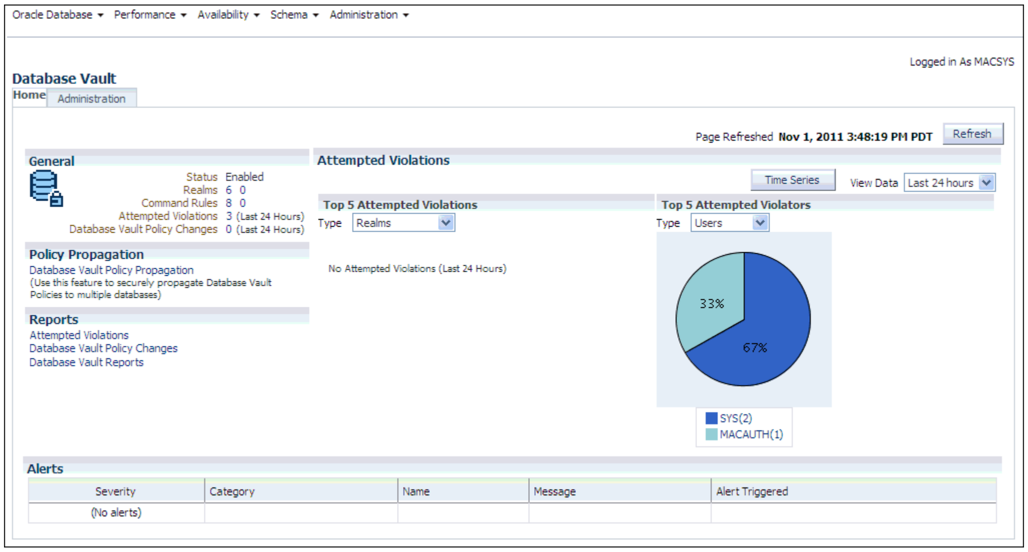
   The Database Login page appears.

8. 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

     See "About Oracle Database Vault Roles" on page 12-3 for more information.
   - **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.



Logged in As MACSYS

Database Vault

Page Refreshed **Nov 1, 2011 3:48:19 PM PDT**

General

Status	Enabled	
Realms	6	0
Command Rules	8	0
Attempted Violations	3	(Last 24 Hours)
Database Vault Policy Changes	0	(Last 24 Hours)

Policy Propagation

Database Vault Policy Propagation
(Use this feature to securely propagate Database Vault Policies to multiple databases)

Reports

Attempted Violations
Database Vault Policy Changes
Database Vault Reports

Attempted Violations

View Data: Last 24 hours

Top 5 Attempted Violations

Type: Realms

No Attempted Violations (Last 24 Hours)

Top 5 Attempted Violators

Type: Users

SYS(2)
MACAUTH(1)

Alerts

Severity	Category	Name	Message	Alert Triggered
(No alerts)				

See Also: "Using Oracle Database Vault with Oracle Enterprise Manager" on page 11-1

Quick Start Tutorial: Securing a Schema from DBA Access

This tutorial shows how to create a realm around the HR schema. After you create the realm, you will test the realm with a user account.

Topics:

- About This Tutorial
- Step 1: Log On as SYSTEM to Access the HR Schema
- Step 2: Create a Realm
- Step 3: Create the SEBASTIAN User Account
- Step 4: Create an Authorization for the Realm
- Step 5: Test the Realm
- Step 6: If Unified Auditing Is Not Enabled, Then Run a Report
- Step 7: Remove the Components for This Tutorial

About This Tutorial

In this tutorial, you create a simple security configuration for the HR sample database schema.

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.

Before you can use this tutorial, ensure that the HR sample schema is installed. See *Oracle Database Sample Schemas* for information on installing the sample schemas.

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

You must enable the HR schema for this tutorial.

1. Log into the database instance as user SYSTEM and access the HR schema.

   ```
   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, query the DBA_PDBS data dictionary view. 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 CDBs: sqlplus bea_dvacctmgr@hrpdb
   Enter password: password

   ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password
   ```

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

   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. Your first step is to create a realm for the HR schema.

1. Log into 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 into Oracle Database Vault" on page 3-7 explains how to log in.
2. In the Administration page, under Database Vault Components, click **Realms**. (It should be selected by default.)
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. After Status, ensure that **Enabled** is selected so that the realm can be used.
7. Under Audit Options, ensure that **Audit On Failure** is selected so that you can create an audit trial later on.
8. Click **Next** to display the Realm secured objects page.
9. 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.
10. Click **OK**.

    The HR.EMPLOYEES table is added to the Create Realm : Realm Secured Objects page.
11. 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 5: Test the Realm, when you test the realm.

Step 4: Create an Authorization for the Realm

At this stage, even though `SEBASTIAN` has the `SELECT ANY TABLE` privilege, he cannot select from the `HR.EMPLOYEES` table because it is protected by a realm.

Next, authorize user `SEBASTIAN` to have access to the HR Apps realm as follows:

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 (0) 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 5: Test the Realm

To test the realm, try accessing 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.

In SQL*Plus, connect as `SYSTEM`, and then try accessing the salary information in the `EMPLOYEES` table again:

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

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. Try the following:

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

```
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 6: 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 such as the one you attempted in Step 5: 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 7: Remove the Components for This Tutorial" on page 3-13.

   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 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. 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 the Database Vault Home page, click **Administration**.

   b. In the Realms page, select `HR Apps` from the list of realms.

   c. Click **Delete**, and in the Confirmation window, click **Yes**.

3. If necessary, lock and expire the `HR` account.

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

4 Performing Privilege Analysis to Find Privilege Use

Privilege analysis enables you to analyze the privileges that users use, and then revoke and regrant these privileges as necessary.

Topics:

- About Privilege Analysis
- Creating and Managing Privilege Analysis Policies
- Creating Roles and Managing Privileges Using Cloud Control
- Tutorial: Analyzing ANY Privilege Use
- Tutorial: Analyzing Privilege Use by a User Who Has the DBA Role
- Privilege Analysis Policy and Report Data Dictionary Views

What Is Privilege Analysis?

Privilege analysis works with a variety of pre-compiled database objects, such as functions, and provides many benefits to better manage your users' privilege use. Only users who have been granted the appropriate privileges can run privilege analysis reports.

Topics:

- About Privilege Analysis
- How Privilege Analysis Works with Pre-Compiled Database Objects
- Who Can Perform Privilege Analysis?
- Types of Privilege Analysis
- Benefits and Use Cases of Privilege Analysis
- How Does a Multitenant Environment Affect Privilege Analysis?

About Privilege Analysis

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

Privilege analysis captures privileges used by database users and applications at runtime. 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.

Privilege analysis support is available from Oracle Enterprise Manager Cloud Control 12*c* Release 3 Plug-in Update 1 (12.1.0.3).

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 such as 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.
- **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 and in developing secure applications.

Topics:

- Unnecessarily Granted Privileges of Applications
- Development of Secure Applications

Unnecessarily Granted Privileges of Applications

When an application accesses a database, the privileges of the account that is used to access the 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

When applications are being developed, some security administrators initially grant many powerful system privileges and roles to application developers because at that stage they (the administrators) 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 used privilege analysis policies in a multitenant environment.

If you are using a multitenant environment, each privilege analysis policy only analyzes and reports privileges exercised within the pluggable database (PDB) where the privilege analysis policy resides.

> **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.

Topics:

- About Creating and Managing Privilege Analysis Policies
- General Steps for Managing Privilege Analysis
- Creating a Privilege Analysis Policy
- Examples of Privilege Analysis Policies
- Enabling a Privilege Analysis Policy
- Disabling a Privilege Analysis Policy
- Generating a Privilege Analysis Report
- Dropping a Privilege Analysis Policy

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 follows a general 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.

3. Disable the privilege analysis policy's recording of privilege use.

   This step stops capturing the privilege use for the policy.

4. Generate privilege analysis results.

   This step writes the results to the data dictionary views described in "Privilege Analysis Policy and Report Data Dictionary Views" on page 4-25.

5. Optionally, disable and then drop the privilege analysis policy.

   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.

Topics:

- About Creating a Privilege Analysis Policy
- Creating a Privilege Analysis Policy in Enterprise Manager Cloud Control
- Creating a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE

About Creating a Privilege Analysis Policy

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

However, the user who created the policy can drop it, as well as user `SYS`. After you create the policy, you must manually enable it so that it can begin to analyze privilege use. If you want to use Oracle Enterprise Manager Cloud Control, then you must use Enterprise Manager Release 12.1.0.3 or later.

Creating a Privilege Analysis Policy in Enterprise Manager Cloud Control

You can create a privilege analysis policy in Cloud Control.

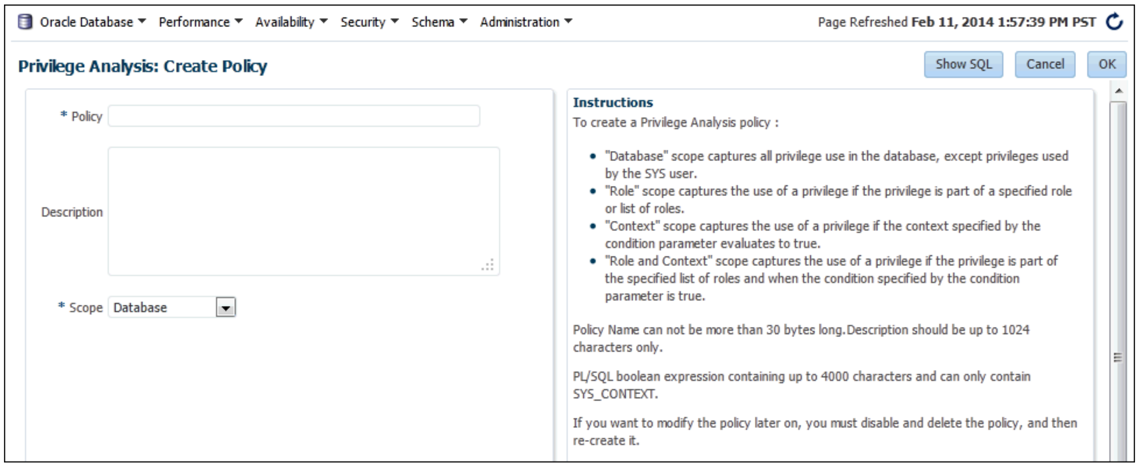
1. In Enterprise Manager, access the target Database home page as a user who has been granted the `CAPTURE_ADMIN` role and the `SELECT ANY DICTIONARY` privilege.

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

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

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

   The Privilege Analysis: Create Policy page appears.



Page Refreshed **Feb 11, 2014 1:57:39 PM PST**

Privilege Analysis: Create Policy

* Policy

Description

* Scope Database

Instructions

To create a Privilege Analysis policy :

- "Database" scope captures all privilege use in the database, except privileges used by the SYS user.
- "Role" scope captures the use of a privilege if the privilege is part of a specified role or list of roles.
- "Context" scope captures the use of a privilege if the context specified by the condition parameter evaluates to true.
- "Role and Context" scope captures the use of a privilege if the privilege is part of the specified list of roles and when the condition specified by the condition parameter is true.

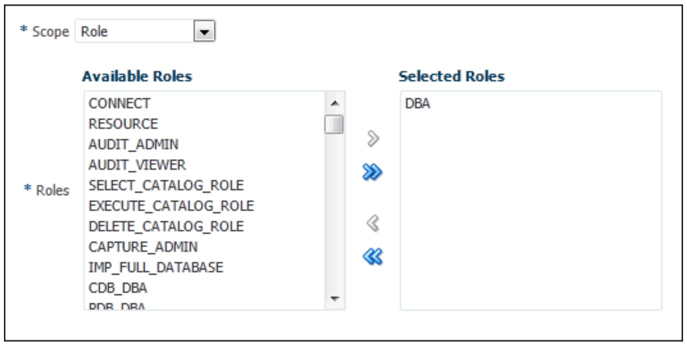
Policy Name can not be more than 30 bytes long.Description should be up to 1024 characters only.

PL/SQL boolean expression containing up to 4000 characters and can only contain SYS_CONTEXT.

If you want to modify the policy later on, you must disable and delete the policy, and then re-create it.

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.



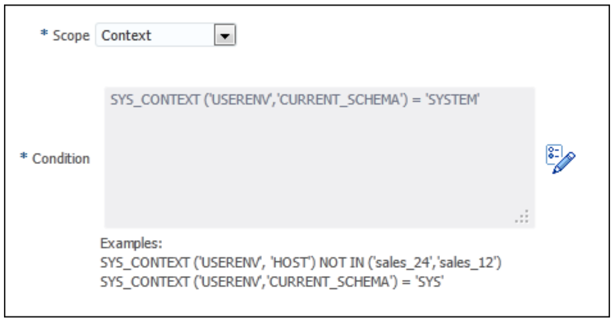
* Scope Role

* Roles

Available Roles: CONNECT, RESOURCE, AUDIT_ADMIN, AUDIT_VIEWER, SELECT_CATALOG_ROLE, EXECUTE_CATALOG_ROLE, DELETE_CATALOG_ROLE, CAPTURE_ADMIN, IMP_FULL_DATABASE, CDB_DBA, PDB_DBA

Selected Roles: DBA

     - **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.



SYS_CONTEXT ('USERENV','CURRENT_SCHEMA') = 'SYSTEM'

Examples:
SYS_CONTEXT ('USERENV', 'HOST') NOT IN ('sales_24','sales_12')
SYS_CONTEXT ('USERENV','CURRENT_SCHEMA') = 'SYS'

– **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**.

Creating a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE

To create a privilege analysis policy using the DBMS_PRIVILEGE_CAPTURE package, you can use the DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE procedure.

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 that is defined by the condition parameter. If you omit this 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" on page 4-10.

Examples of Privilege Analysis Policies

You can create a variety of privilege analysis policies.

Topics:

- Example: Privilege Analysis of Database-Wide Privileges
- Example: Privilege Analysis of Privilege Usage of Two Roles
- Example: Privilege Analysis of Privileges During SQL*Plus Use
- Example: Privilege Analysis of PSMITH Privileges During SQL*Plus Access

Example: Privilege Analysis of 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 ('all_privs_capture');
```

Example: Privilege Analysis of Privilege Usage of Two 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');
```

Example: Privilege Analysis of Privileges During SQL*Plus Use

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');
```

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

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');
```

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.

Topics:

- About Enabling a Privilege Analysis Policy
- Enabling a Privilege Analysis Policy Using Cloud Control
- Enabling a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE

About Enabling a Privilege Analysis Policy

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

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.

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.

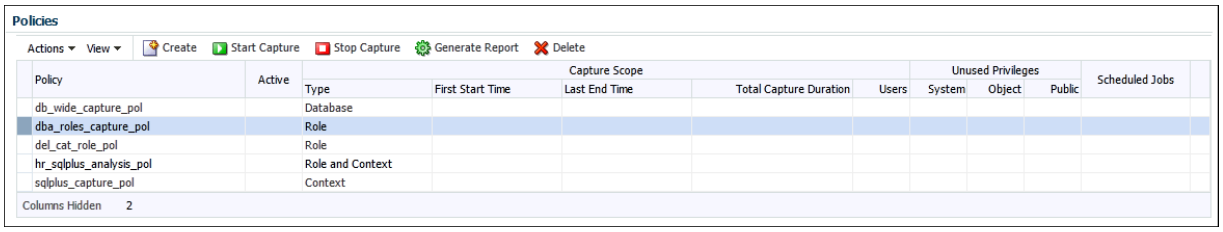
Enabling a Privilege Analysis Policy Using Cloud Control

You can enable a privilege analysis policy using Cloud Control.

1. 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.

2. From the **Security** menu, select **Privilege Analysis**.
3. Under Policies, select the policy that you want to enable.



Policies

Policy	Active	Capture Scope					Unused Privileges			Scheduled Jobs
		Type	First Start Time	Last End Time	Total Capture Duration	Users	System	Object	Public	
db_wide_capture_pol		Database								
dba_roles_capture_pol		Role								
del_cat_role_pol		Role								
hr_sqlplus_analysis_pol		Role and Context								
sqlplus_capture_pol		Context								

Columns Hidden 2

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

You can enable a privilege policy by using the `DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE` procedure.

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.

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

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

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.

Topics:

- About Disabling a Privilege Analysis Policy
- Disabling a Privilege Analysis Policy Using Cloud Control
- Disabling a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE

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. 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.

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, specify a time to stop the privilege analysis policy.

   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.
6. Click **OK**.

Disabling a Privilege Analysis Policy Using DBMS_PRIVILEGE_CAPTURE

You can use the `DBMS_PRIVILEGE_CAPTURE.DISABLE_CAPTURE` procedure to disable 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.

Topics:

- About Generating a Privilege Analysis Report
- Generating a Privilege Analysis Report Using Cloud Control
- Accessing Privilege Analysis Reports Using Cloud Control
- Generating a Privilege Analysis Report Using DBMS_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" on page 4-25. 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. 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.

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.

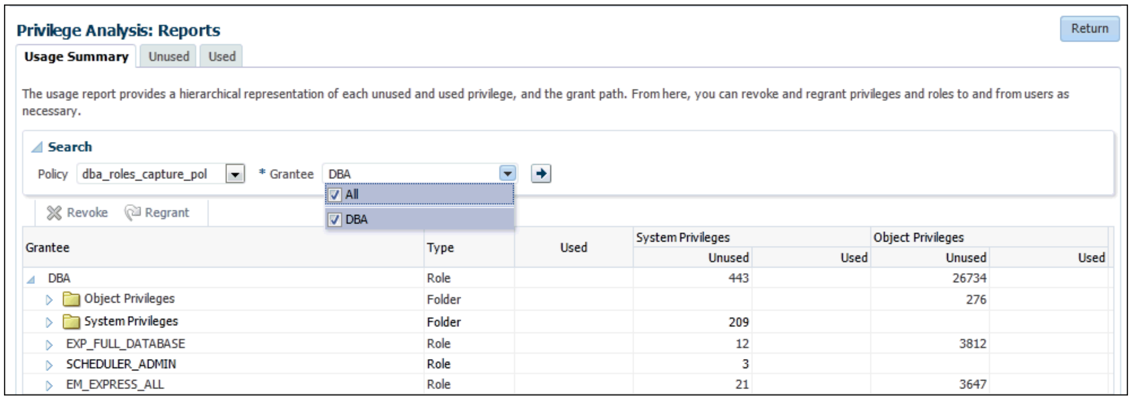
Accessing Privilege Analysis Reports Using Cloud Control

After you have generated a privilege analysis report in Cloud Control, you can access the report.

1. Generate the privilege analysis report.

   See "Generating a Privilege Analysis Report Using Cloud Control" on page 4-12 for more information.
2. In the Privilege Analysis page, select the policy on which you generated a report.
3. From the **Actions** menu, select **Reports**.

   The Privilege Analysis Reports page appears. The following image shows the **Usage Summary** tab, with the **Search** field expanded. It also shows how many system privileges not used.



The usage report provides a hierarchical representation of each unused and used privilege, and the grant path. From here, you can revoke and regrant privileges and roles to and from users as necessary.

Policy dba_roles_capture_pol * Grantee DBA
☑ All
☑ DBA

Grantee	Type	Used	System Privileges		Object Privileges	
			Unused	Used	Unused	Used
DBA	Role		443		26734	
Object Privileges	Folder				276	
System Privileges	Folder		209			
EXP_FULL_DATABASE	Role		12		3812	
SCHEDULER_ADMIN	Role		3			
EM_EXPRESS_ALL	Role		21		3647	

4. Select from the **Usage Summary**, **Unused**, and **Used** tabs to find detailed information about the privilege use that was found by the policy.

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

Generating a Privilege Analysis Report Using DBMS_PRIVILEGE_CAPTURE

You can generate a report showing the results of the privilege capture.

1. Run the `DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT` procedure.

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

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

2. 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.

Topics:

- About Dropping a Privilege Analysis Policy
- Dropping a Privilege Analysis Policy Using Cloud Control
- Dropping a Privilege Analysis Policy Using the DBMS_PRIVILEGE_CAPTURE 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 also drops all the used and unused privilege records associated with this privilege analysis.

Dropping a Privilege Analysis Policy Using Cloud Control

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

1. 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.
2. From the **Security** menu, select **Privilege Analysis**.
3. Under Policies, select the policy that you want to drop.
4. Select **Delete**.
5. In the Confirmation dialog box, select **Yes**.

Dropping a Privilege Analysis Policy Using the DBMS_PRIVILEGE_CAPTURE Package

You can drop a privilege analysis policy in SQL*Plus by using the `DBMS_PRIVILEGE_CAPTURE` PL/SQL package.

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');
   ```

Creating Roles and Managing Privileges Using Cloud Control

Based on findings from a privilege analysis report in Enterprise Manager Cloud Control, you can create a new role using the privileges found, and the grant this role to users.

Topics:

- Creating a Role from a Privilege Analysis Report in Cloud Control
- Revoking and Regranting Roles and Privileges Using Cloud Control
- Generating a Revoke or Regrant Script Using Cloud Control

Creating a Role from a Privilege Analysis Report in Cloud Control

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

1. Access the Privilege Analysis page.

   See "Accessing Privilege Analysis Reports Using Cloud Control" on page 4-13 for more information.

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.

     – **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.

     – **None** for role, then no role that is captured in the policy will be used in the new 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.

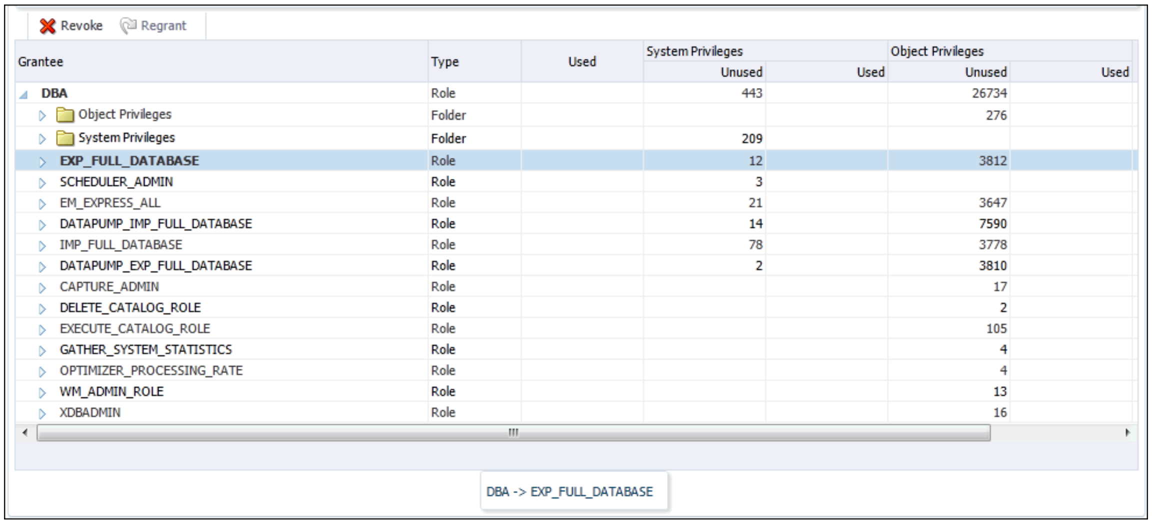
   In SQL*Plus, a user who has been granted the `DV_OWNER` role can check the authorization by querying the `DVSYS.DBA_DV_REALM_AUTH` data dictionary view. To grant the user authorization, use the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure.

2. Access the Privilege Analysis Reports page.

In the Privilege Analysis page, select the policy that is associated with the report, and then from the **Actions** menu, select **Reports**.

3. In the Privilege Analysis Reports page, expand the grantee whose privileges you want to revoke or regrant.

   The following image shows the `EXP_FULL_DATABASE` role, which can be revoked.



Grantee	Type	Used	System Privileges		Object Privileges	
			Unused	Used	Unused	Used
DBA	Role		443		26734	
Object Privileges	Folder				276	
System Privileges	Folder		209			
EXP_FULL_DATABASE	Role		12		3812	
SCHEDULER_ADMIN	Role		3			
EM_EXPRESS_ALL	Role		21		3647	
DATAPUMP_IMP_FULL_DATABASE	Role		14		7590	
IMP_FULL_DATABASE	Role		78		3778	
DATAPUMP_EXP_FULL_DATABASE	Role		2		3810	
CAPTURE_ADMIN	Role				17	
DELETE_CATALOG_ROLE	Role				2	
EXECUTE_CATALOG_ROLE	Role				105	
GATHER_SYSTEM_STATISTICS	Role				4	
OPTIMIZER_PROCESSING_RATE	Role				4	
WM_ADMIN_ROLE	Role				13	
XDBADMIN	Role				16	

4. Select a category under this user name, such as a specific role, or the privileges listed in the **System Privileges** or **Object Privileges** folders.

   The **Revoke** button is enabled if the role or privilege is currently granted to the user. If it is not, then the **Regrant** button is enabled.

5. To revoke the role or privilege, select **Revoke**; to regrant it, select **Regrant**.

6. In the Confirmation window, select **Yes**.

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.

Topics:

- About Generating Revoke and Regrant Scripts
- Generating a Revoke Script
- Generating a Regrant Script

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.

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 `DVSYS.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" on page 4-12 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, select a policy name from the menu against which the revoke script needs to be prepared.

8. Enter a unique name and description for the script.

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

   On the Review page, you can see a list of all the privileges that are going to be included in the revoke script.

10. Click **Save**.

    A Confirmation window appears.

11. In the Revoke Scripts page, find the newly created SQL script, and then click **Revoke Script** to download this script.

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 `DVSYS.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" on page 4-12 for more information.

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

6. In the Revoke Scripts page, find the regrant script that corresponds to the revoke script that you had generated earlier, and then click **Regrant Script** to download this script.

Tutorial: Analyzing ANY Privilege Use

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

Topics:

- Step 1: Create User Accounts
- Step 2: Create and Enable a Privilege Analysis Policy
- Step 3: Use the READ ANY TABLE System Privilege
- Step 4: Disable the Privilege Analysis Policy
- Step 5: Generate and View a Privilege Analysis Report
- Step 6: Remove the Components for This Tutorial

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, connect to the appropriate 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 `DVSYS.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.

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

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.

```
EXEC DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT ('ANY_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" on page 4-25.

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
```

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

```
SELECT USERNAME, SYS_PRIV, OBJECT_OWNER, OBJECT_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.

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

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 `ANY_priv_analysis_pol` privilege analysis policy.

   ```
   EXEC DBMS_PRIVILEGE_CAPTURE.DROP_CAPTURE ('ANY_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 `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 system and object privilege use of a user who has been granted the `DBA` role and who performs database tuning operations.

Topics:

- Step 1: Create User Accounts
- Step 2: Create and Enable a Privilege Analysis Policy
- Step 3: Perform the Database Tuning Operations
- Step 4: Disable the Privilege Analysis Policy
- Step 5: Generate and View Privilege Analysis Reports
- Step 6: Remove the Components for This Tutorial

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, log into the appropriate 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;
```

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 `DVSYS.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` is ready to 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" on page 4-25.

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

Table 4–1 lists data dictionary views that you can use to find information about analyzed privileges.

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 that have been used for reported privilege analysis policies
`DBA_UNUSED_PRIVS`	Lists the privileges that have not been used for reported privilege analysis policies
`DBA_USED_OBJPRIVS`	Lists the object privileges that have been used for reported privilege analysis policies. It does not include the object grant paths.
`DBA_UNUSED_OBJPRIVS`	Lists the object privileges 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 that have been used for reported privilege analysis policies. It includes the object privilege grant paths.
`DBA_UNUSED_OBJPRIVS_PATH`	Lists the object privileges that have not been used for reported privilege analysis policies. It includes the object privilege grant paths.
`DBA_USED_SYSPRIVS`	Lists the system privileges 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 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 that have been used for reported privilege analysis policies. It includes the system privilege grant paths.
`DBA_UNUSED_SYSPRIVS_PATH`	Lists the system privileges that have not been used for reported privilege analysis policies. It includes system privilege grant paths
`DBA_USED_PUBPRIVS`	Lists all the privileges for the `PUBLIC` role that have been used for reported privilege analysis policies
`DBA_USED_USERPRIVS`	Lists the user privileges 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 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 that have been used for reported privilege analysis policies. It includes the user privilege grant paths.
`DBA_UNUSED_USERPRIVS_PATH`	Lists the privileges 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 special authorizations to control the access users have to this data.

Topics:

See Also: Chapter 13, "Oracle Database Vault Realm APIs"

What Are Realms?

Realms enable to protect database objects. A mandatory realm restricts user access to objects within a realm. Realm can protect specific object types.

Topics:

About Realms

A **realm** is a functional grouping of database schemas, database objects, and/or 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.

You can control the access that users have to their own objects inside a realm by configuring it to be a mandatory realm. Mandatory realms block both object privilege-based and system privilege-based access. 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.

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.

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.

You can run reports on realms that you create in Oracle Database Vault. See "Related Reports and Data Dictionary Views" on page 5-16 for more information.

This chapter explains how to configure realms by using the Oracle Database Vault Administrator pages in Oracle Enterprise Manager Cloud Control. To configure realms by using the PL/SQL interfaces and packages provided by Oracle Database Vault, refer to Chapter 13, "Oracle Database Vault Realm APIs."

Using 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.

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.**

  See Also:

  - "CREATE_REALM Procedure" on page 13-5
  - "UPDATE_REALM Procedure" on page 13-9

Object Types That Realms Can Protect

You can create realms around a range of types.

`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 a set of default realms. These realms are regular realms, not mandatory realms.

Topics:

- Oracle Database Vault Realm
- Database Vault Account Management Realm
- Oracle Enterprise Manager Realm
- Oracle Default Schema Protection Realm
- Oracle System Privilege and Role Management Realm
- Oracle Default Component Protection Realm

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 these schemas are owners of this realm. For more information about these schemas, see "Oracle Database Vault Schemas" on page 12-1 and "Oracle Database Vault Accounts" on page 12-19.

This realm protects the following objects:

- Entire schemas that are protected:

DVSYS	DVF	LABACSYS

- Roles that are protected:

DV_ADMIN	DV_PUBLIC	DV_GOLDENGATE_ADMIN
DV_AUDIT_CLEANUP	DV_PATCH_ADMIN	DV_XSTREAM_ADMIN
DV_DATAPUMP_NETWORK_LINK	DV_MONITOR	DV_GOLDENGATE_REDO_ACCESS
DV_OWNER	DV_STREAMS_ADMIN	
DV_SECANALYST	LBAC_DBA	

- PL/SQL package that is protected:

  `SYS.DBMS_RLS`

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.

See "DV_ACCTMGR Database Vault Account Manager Role" on page 12-16 for more information about the `DV_ACCTMGR` role.

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 OLAP, Oracle Spatial, and 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

Three 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.

SYS	CTXSYS	EXFSYS

Oracle System Privilege and Role Management Realm

Oracle System Privilege and Role Management Realm protects all sensitive roles that are used for exporting and importing data to and from 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:

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

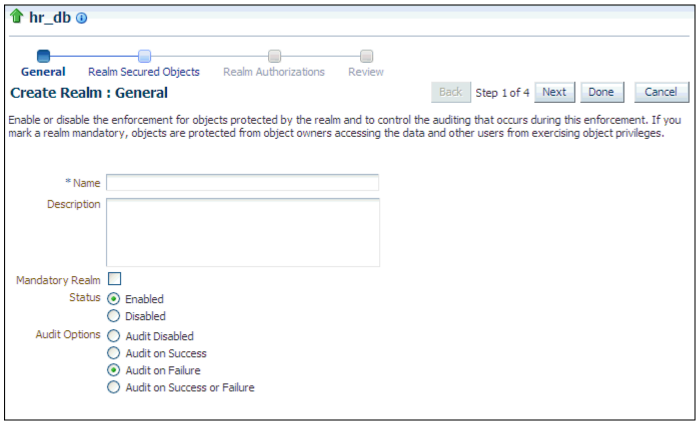
In general, to enable realm protection, you create the realm and configure it to include realm-secured objects, roles, and authorizations.

"Guidelines for Designing Realms" on page 5-15 provides advice on creating realms.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the DV_OWNER or DV_ADMIN role.

   "Logging into Oracle Database Vault" on page 3-7 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.



Create Realm : General

Enable or disable the enforcement for objects protected by the realm and to control the auditing that occurs during this enforcement. If you mark a realm mandatory, objects are protected from object owners accessing the data and other users from exercising object privileges.

* Name

Description

Mandatory Realm ☐

Status [✓ Enabled] Disabled

Audit Options Audit Disabled, Audit on Success, [✓ Audit on Failure], Audit on Success or Failure

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 "Using Mandatory Realms to Restrict User Access to Objects within a Realm" on page 5-2 for more information about mandatory realms.

- **Status:** Select either **Enabled** or **Disabled** to enable or disable the realm. 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 Appendix A, "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*.

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

   See "About Realm-Secured Objects" on page 5-9 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. 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.

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

   See "About Realm Authorization" on page 5-9 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 attribute is mandatory.

     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.

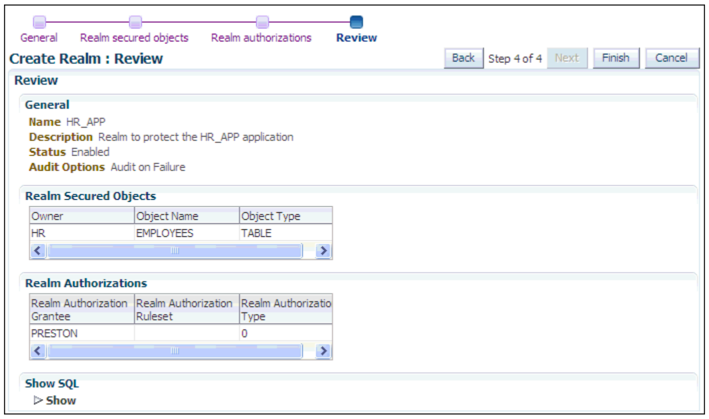
     See "Creating a Rule to Add to a Rule Set" on page 6-6 for more information about defining rules to govern the realm authorization.

     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:



Create Realm : Review

Review

General

Name HR_APP
Description Realm to protect the HR_APP application
Status Enabled
Audit Options Audit on Failure

Realm Secured Objects

Owner	Object Name	Object Type
HR	EMPLOYEES	TABLE

Realm Authorizations

Realm Authorization Grantee	Realm Authorization Ruleset	Realm Authorizatio Type
PRESTON		0

Show SQL

11. Click **Finish** to complete the realm creation.

See Also:

- "About Realm-Secured Objects" on page 5-9
- "About Realm Authorization" on page 5-9
- "Propagating Oracle Database Vault Policies to Other Databases" on page 11-1

About Realm-Secured Objects

Realm-secured objects define the *territory* that a realm protects. The realm territory is a set of schema and database objects and roles.

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.

  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.

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. To track configuration information for the authorization of a realm, see "Realm Authorization Configuration Issues Report" on page 24-4.

Disabling and Enabling a Realm

You can disable or enable a realm from Enterprise Manager Cloud Control.

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** or **Enabled**.
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.

   See Chapter 22 for more information about these views.
2. In the Oracle Database Vault Administration page, select **Realms**.
3. In the Realms page, select the realm you want to delete, and then select **Remove**.
4. 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.

How Realms Work

When a database account that has the appropriate privileges issues a SQL statement (that is, DDL, DML, `EXECUTE`, `GRANT`, `REVOKE`, or `SELECT`) that affects an object within a realm, a special set of activities occur.

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.

   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

Realms authorizations work by checking if users have the privileges that they need to perform a specific task. If the user does not have the correct privileges, then the user cannot access the data.

Topics:

- About Authorizations in a Realm
- Example: Unauthorized User Trying to Create a Table
- Example: Unauthorized User Trying to Use the DELETE ANY TABLE Privilege
- Example: Authorized User Performing DELETE Operation

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 Creating Realms

You can create realms that protect objects from users who have system privileges and other powerful privileges, for example.

Topics:

- Example: Unauthorized User Trying to Create a Table
- Example: Unauthorized User Trying to Use the DELETE ANY TABLE Privilege
- Example: Authorized User Performing DELETE Operation

Example: Unauthorized User Trying to Create a Table

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

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

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

There are situations in which you may want to 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.

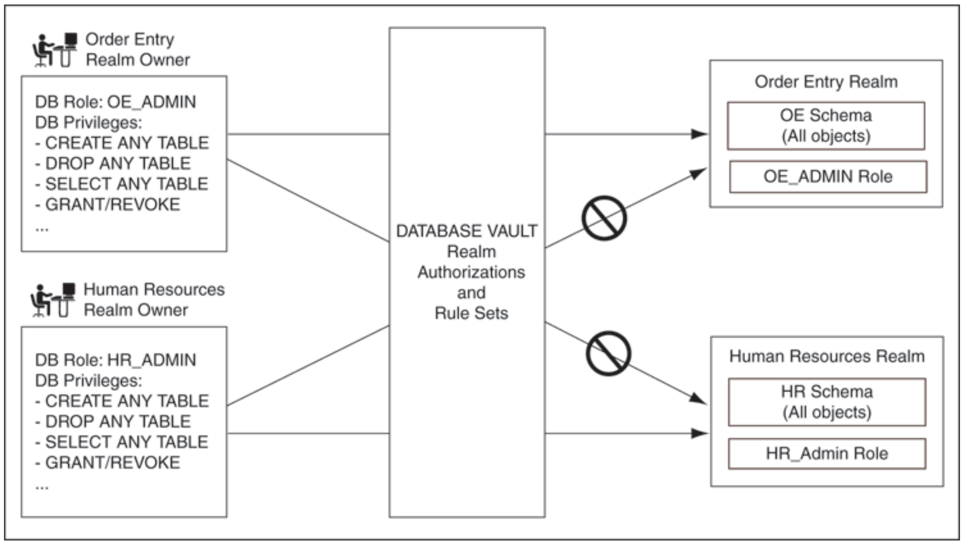
Example of How Realms Work

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



See Also: "Quick Start Tutorial: Securing a Schema from DBA Access" on page 3-9 for a tutorial on how to create and use a realm

How Realms Affect Other Oracle Database Vault Components

Realms have no effect on factors, identities, or rule sets. They have an effect on command rules, in a sense, in that Oracle Database Vault evaluates the realm authorization first when processing SQL statements.

"How Realms Work" on page 5-10 explains the steps that Oracle Database Vault takes to process SQL statements that affect objects in a realm. "How Command Rules Work" on page 7-6 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;
/
```

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`. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Cloud Control, refer to its online Help. See *Oracle Database Performance Tuning Guide* to learn how to monitor database performance, and *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements.

Related Reports and Data Dictionary Views

Table 5–1 lists Oracle Database Vault reports that are useful for analyzing realms. See Chapter 24, "Oracle Database Vault Reports," for information about how to run these reports.

Table 5–1 Reports Related to Realms

Report	Purpose
"Realm Audit Report" on page 24-5	Audits records generated by the realm protection and realm authorization operations
"Realm Authorization Configuration Issues Report" on page 24-4	Lists authorization configuration information, such as incomplete or disabled rule sets, or nonexistent grantees or owners that may affect the realm

Table 5–1 (Cont.) Reports Related to Realms

Report	Purpose
"Rule Set Configuration Issues Report" on page 24-3	Lists rule sets that do not have rules defined or enabled, which may affect the realms that use them
"Object Privilege Reports" on page 24-6	Lists object privileges that the realm affects
"Privilege Management - Summary Reports" on page 24-10	Provides information about grantees and owners for a realm
"Sensitive Objects Reports" on page 24-8	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
"DVSYS.DBA_DV_REALM View" on page 22-14	Lists the realms created in the current database instance.
"DVSYS.DBA_DV_REALM_AUTH View" on page 22-15	lists the authorization of a named database user account or database role (`GRANTEE`) to access realm objects in a particular realm
"DVSYS.DBA_DV_REALM_OBJECT View" on page 22-16	Lists the database schemas, or subsets of schemas with specific database objects contained therein, that are secured by the realms

6 Configuring Rule Sets

You can use rule sets to group a set of rules together. The rules determine whether a user can perform an action on an object.

Topics:

See Also: Chapter 14, "Oracle Database Vault Rule Set APIs"

What Are Rule Sets?

A **rule set** is a collection of one or more rules that you can associate 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.

You can run reports on the rule sets that you create in Oracle Database Vault. See "Related Reports and Data Dictionary Views" on page 6-26 for more information.

This chapter explains how to configure rule sets by using Oracle Database Vault Administrator. To configure rule sets by using the PL/SQL interfaces and packages provided by Oracle Database Vault, refer to Chapter 14, "Oracle Database Vault Rule Set APIs."

Default Rule Sets

Oracle Database Vault provides a set of default rules sets that you can customize for your needs.

The default rule sets are as follows:

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

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), cannot set `UTL_FILE_DIR`
  - **Are Security Parameters Allowed rule:** Cannot set the following:

AUDIT_SYS_OPERATIONS = FALSE	OS_ROLES = TRUE
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 System Parameters:** 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.
- **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.
- **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" on page 12-16 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.

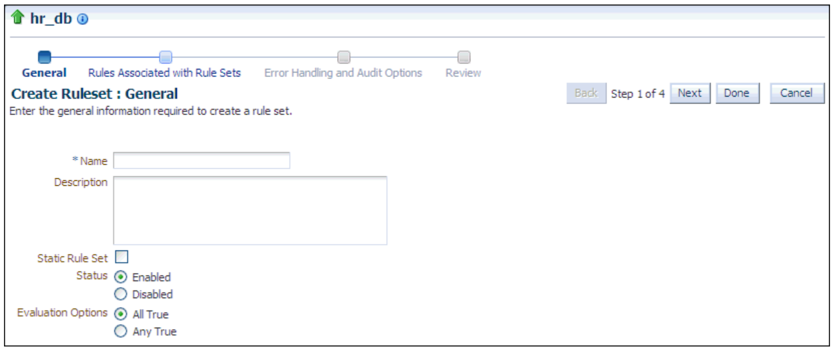
Creating a Rule Set

In general, to create a rule set, you first create the rule set itself, and then you 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.

To create a rule set:

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 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.


hr_db
General
Rules Associated with Rule Sets
Error Handling and Audit Options
Review
Back
Step 1 of 4
Next
Done
Cancel

Create Ruleset : General

Enter the general information required to create a rule set.

* Name

Description

Static Rule Set ☐

Status [✓ Enabled] Disabled

Evaluation Options [✓ All True] Any True

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.

     – **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.

5. Click **Next** to display the Rules Associated with Rule Sets 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" on page 6-6 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 Chapter 14, "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:

- "Guidelines for Designing Rule Sets" on page 6-24 for advice on designing rule sets
- "Oracle Database Vault PL/SQL Rule Set Functions" on page 14-11 for a set of functions that you can use in rule expressions
- "Rule Set Configuration Issues Report" on page 24-3 to check the configuration of the rule sets for your database
- "Propagating Oracle Database Vault Policies to Other Databases" on page 11-1
- Appendix A, "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. You can create one or more related rules and then add them to a named collection of rules called a rule set.

Topics:

- About Creating Rules
- Default Rules
- Creating a New Rule
- Adding Existing Rules to a Rule Set

About Creating Rules

You can create rules during the rule set creation process, or independently of it. You can associate the 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" on page 6-24 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.

See "How Rule Sets Work" on page 6-12 for information on how rules are evaluated, how to nest rules, and how to create rules that exclude a particular user, such as a privileged user.

Default Rules

You can find a full list of rules by querying the `DVSYS.DBA_DV_RULE` data dictionary view.

Table 6–1 lists the default Oracle Database rules.

Table 6–1 Default Oracle Database Vault Rules

Rule	Description
Are Backup Restore Parameters Allowed	Checks if the current SQL statement attempts to turn on the `RECYCLEBIN` parameter
Are Database File Parameters Allowed	Checks if the current SQL statement attempts to alter control file related configuration
Are Dump or Dest Parameters Allowed	Checks if the current SQL statement attempts to alter initialization parameters related to the size limit or destination of dump
Are Optimizer Parameters Allowed	Checks if the current SQL statement attempts to alter the setting for the `OPTIMIZER_SECURE_VIEW_MERGING` parameter
Are PL-SQL Parameters Allowed	Checks if the current SQL statement attempts to alter the following initialization parameters: ■ `UTL_FILE_DIR` ■ `PLSQL_DEBUG`
Are Security Parameters Allowed	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`
Are System Security Parameters Allowed	Prevents modification of the following parameters: ■ `07_DICTIONARY_ACCESSIBILITY` ■ `DYNAMIC_RLS_POLICIES` ■ `_SYSTEM_TRIG_ENABLED`
False	Evaluates to `FALSE`

Table 6–1 (Cont.) Default Oracle Database Vault Rules

Rule	Description
Is Alter DVSYS Allowed	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 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	Prevent modification of the `DYNAMIC_RLS_POLICIES` parameter
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
Is User Manager	Checks if a user has been granted the `DV_ACCTMGR` role
Is _system_trig_enabled Parameters Allowed	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` ■ `UTL_FILE_DIR`: Blocks all operations on this parameter Note that if you have enabled unified auditing, then the `AUDIT_SYS_OPERATIONS`, `AUDIT_TRAIL`, and `AUDIT_SYSLOG_LEVEL` parameters have no effect.

Table 6–1 (Cont.) Default Oracle Database Vault Rules

Rule	**Description**
Is o7_dictionary_accessibility Parameters Allowed	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.

To create and add a rule to a rule set:

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 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;
```

See the following sections for functions that you can use in the rule set expression:

– "Oracle Database Vault PL/SQL Rule Set Functions" on page 14-11
– "DBMS_MACADM Rule Set Procedures" on page 14-1
– Chapter 19, "Oracle Database Vault Utility APIs"

For additional examples of expressions, see the rule defined in the rule sets provided with Oracle Database Vault. "Default Rule Sets" on page 6-2 lists these rule sets.

5. Click **OK**.

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.

To add existing rules to a rule set:

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 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 Rules Associated with Rule Sets 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 by querying the rules-related Oracle Database Vault views.

See Chapter 22 for more information about Oracle Database Vault views.

To remove a rule from a rule set:

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 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 Rules Associated with Rule Sets 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, 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. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 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 **Remove**.
7. Click **Done**, then click **Finish**.

Deleting a Rule Set

You can use Enterprise Manager Cloud Control to delete a rule set.

1. Ensure that there are no references to the rule set.

   Before you can delete a rule set, you must remove references to this rule set.

   See "Removing Rule Set References to Oracle Database Vault Components" on page 6-11.

2. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 explains how to log in.

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.

How Rule Sets Work

It is important to understand how Oracle Database Vault evaluates rule sets, the nesting rules within a rule set, and how to create rules that apply to everyone except one users.

- How Oracle Database Vault Evaluates Rules
- Nested Rules within a Rule Set
- Creating Rules to Apply to Everyone Except One 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'
```

You can create it using a factor function. See "Oracle Database Vault DVF PL/SQL Factor Functions" on page 16-24 for more information. Chapter 8 explains how to create factors.

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 create an email alert for security violations. You must use the UTL_MAIL PL/SQL package and an access control list to create this kind of alert.

Topics:

About This Tutorial

In the following 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.

Step 1: Install and Configure the UTL_MAIL PL/SQL Package

The UTL_MAIL PL/SQL package contains a set of procedures that enable you to manage email notifications. You must manually install this package.

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, query the DBA_PDBS data dictionary view. 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 PL/SQL network utility packages such as `UTL_MAIL`, you must configure an access control list (ACL) file that enables fine-grained access to external network services.

For detailed information about this topic, 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" on page 6-13. 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;
```

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

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.

Topics:

About This Tutorial

In this tutorial, you configure a rule set that defines two-person integrity (TPI), 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
   ```

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

   For example:

   ```
   CONNECT 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.

3. 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;
```

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

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

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

5. 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 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 the Users' 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" on page 14-1 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" on page 6-13 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

In general, the more rules and more complex the rules, the more performance overhead the performance for execution of certain operations governed by these rule sets.

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" on page 6-24 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`. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Cloud Control, refer to its online Help. See *Oracle Database Performance Tuning Guide* to learn how to monitor database performance, and *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements.

Related Reports and Data Dictionary Views

Table 6–2 lists Oracle Database Vault reports that are useful for analyzing rule sets and the rules within them. See Chapter 24, "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" on page 24-3	Lists rule sets that have no rules defined or enabled
"Secure Application Configuration Issues Report" on page 24-4	Lists secure application roles that have incomplete or disabled rule sets
"Command Rule Configuration Issues Report" on page 24-3	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
"DVSYS.DBA_DV_RULE View" on page 22-17	Lists the rules that have been defined
"DVSYS.DBA_DV_RULE_SET View" on page 22-17	Lists the rules sets that have been created
"DVSYS.DBA_DV_RULE_SET View" on page 22-17	Lists rules that are associated with existing rule sets

7

Configuring Command Rules

You can create command rules to protect DDL and DML statements. Oracle Database Vault provides a set of default command rules.

Topics:

See Also: Chapter 15, "Oracle Database Vault Command Rule APIs"

What Are Command Rules?

You can create custom command rules or use the default command rules. You can create and use command rules in a multitenant environment.

Topics:

About Command Rules

A **command rule** is a rule that you create to protect `SELECT`, `ALTER SYSTEM`, database definition language (DDL), and data manipulation language (DML) statements that affect one or more database objects.

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. If you want to protect realm-specific objects, see "About Realm Authorization" on page 5-9.

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

For more information about SQL statements and operations, refer to *Oracle Database SQL Language Reference*. See also "SQL Statements That Can Be Protected by Command Rules" on page 7-3.

Command rules can be categorized as follows:

- **Command rules that have a system-wide scope.** With this type, you can only create one command rule for each database instance. Examples are command rules for the `ALTER SYSTEM` and `CONNECT` statements.
- **Command rules that are schema specific.** An example is creating a command rule for the `DROP TABLE` statement.
- **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. Chapter 6, "Configuring Rule Sets," describes rule sets in detail.

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. See "Related Reports and Data Dictionary View" on page 7-10 for more information.

This chapter explains how to configure command rules by using Oracle Database Vault Administrator. To configure command rules by using the `DBMS_MACADM` PL/SQL package, refer to Chapter 15, "Oracle Database Vault Command Rule APIs."

Using Command Rules in a Multitenant Environment

In a multitenant environment, you can create command rules for the `CREATE PLUGGABLE DATABASE`, `ALTER PLUGGABLE DATABASE`, and `DROP PLUGGABLE DATABASE` statements.

To apply these command rules to the entire multitenant environment, create them in the root.

Default Command Rules

Table 7–1 lists default command rules that Oracle Database Vault provides.

Table 7–1 Default Command Rules

SQL Statement	Object Name	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.

SQL Statements That Can Be Protected by Command Rules

You can protect the a large number of SQL statements by using command rules.

The SQL statements that you can protect are as follows:

`ALTER CLUSTER`	`CREATE EDITION`	`DROP DIMENSION`
`ALTER DIMENSION`	`CREATE USER`	`DROP DIRECTORY`
`ALTER FUNCTION`	`CREATE CLUSTER`	`DROP FUNCTION`
`ALTER INDEX`	`CREATE CONTEXT`	`DROP INDEX`
`ALTER INDEXTYPE`	`CREATE DATABASE LINK`	`DROP INDEXTYPE`
`ALTER JAVA`	`CREATE DIMENSION`	`DROP JAVA`
`ALTER LIBRARY`	`CREATE DIRECTORY`	`DROP LIBRARY`
`ALTER OPERATOR`	`CREATE FUNCTION`	`DROP OPERATOR`
`ALTER OUTLINE`	`CREATE INDEX`	`DROP OUTLINE`

ALTER MATERIALIZED VIEW	CREATE INDEXTYPE	DROP MATERIALIZED VIEW
ALTER MATERIALIZED VIEW LOG	CREATE JAVA	DROP MATERIALIZED VIEW LOG
ALTER PACKAGE	CREATE LIBRARY	DROP PACKAGE
ALTER PACKAGE BODY	CREATE OPERATOR	DROP PACKAGE BODY
ALTER PLUGGABLE DATABASE	CREATE OUTLINE	DROP PROCEDURE
ALTER PROCEDURE	CREATE PACKAGE	DROP PLUGGABLE DATABASE
ALTER PROFILE	CREATE PACKAGE BODY	DROP PROFILE
ALTER RESOURCE COST	CREATE PLUGGABLE DATABASE	DROP ROLE
ALTER ROLE	CREATE PROCEDURE	DROP ROLLBACK SEGMENT
ALTER ROLLBACK SEGMENT	CREATE PROFILE	DROP SEQUENCE
ALTER SEQUENCE	CREATE ROLE	DROP SYNONYM
ALTER SESSION	CREATE ROLLBACK SEGMENT	DROP TABLE
ALTER SYNONYM	CREATE SCHEMA	DROP TABLESPACE
ALTER SYSTEM	CREATE SEQUENCE	DROP TRIGGER
ALTER TABLE	CREATE MATERIALIZED VIEW	DROP TYPE
ALTER TABLESPACE	CREATE MATERIALIZED VIEW LOG	DROP TYPE BODY
ALTER TRIGGER	CREATE SYNONYM	DROP USER
ALTER TYPE	CREATE TABLE	DROP VIEW
ALTER TYPE BODY	CREATE TABLESPACE	EXECUTE
ALTER USER	CREATE TRIGGER	GRANT
ALTER VIEW	CREATE TYPE	INSERT
ANALYZE CLUSTER	CREATE TYPE BODY	NOAUDIT
ANALYZE INDEX	CREATE VIEW	RENAME
ANALYZE TABLE	DELETE	REVOKE
ASSOCIATE STATISTICS	DISASSOCIATE STATISTICS	SELECT
AUDIT	DROP CLUSTER	TRUNCATE CLUSTER
CHANGE PASSWORD	DROP CONTEXT	TRUNCATE TABLE
COMMENT	DROP DATABASE LINK	UPDATE
CONNECT	DROP EDITION	

See Also: "Using Command Rules in a Multitenant Environment" on page 7-2 for information about using the `CREATE PLUGGABLE DATABASE`, `ALTER PLUGGABLE DATABASE`, and `DROP PLUGGABLE DATABASE` in a multitenant container database (CDB)

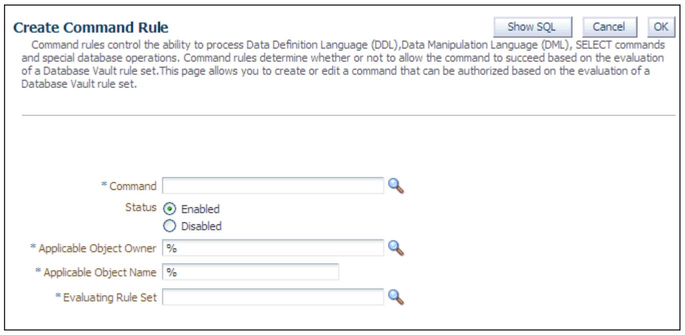
Creating or Editing a Command Rule

You can create or edit a command rule in Oracle Database Vault Administrator.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 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.



Create Command Rule

Show SQL | Cancel | OK

Command rules control the ability to process Data Definition Language (DDL),Data Manipulation Language (DML), SELECT commands and special database operations. Command rules determine whether or not to allow the command to succeed based on the evaluation of a Database Vault rule set.This page allows you to create or edit a command that can be authorized based on the evaluation of a Database Vault rule set.

* Command	
Status	[✓ Enabled] Disabled
* Applicable Object Owner	%
* Applicable Object Name	%
* Evaluating Rule Set	

   - To edit an existing command rule, select it from the list and then click **Edit**.
4. In the Create (Edit) 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** or **Disabled** to enable or disable 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" on page 7-3 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 "Related Reports and Data Dictionary View" on page 7-10 for more information.
   - **Evaluating 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 Chapter 24.) Any auditing and custom event handling associated with the rule set occurs as a part of the command rule processing.

     See Chapter 6, "Configuring Rule Sets," for more information about rule sets.
5. Click **OK**.

See Also: "Propagating Oracle Database Vault Policies to Other Databases" on page 11-1

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. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 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**.

See Also: Appendix 22, "Oracle Database Vault Data Dictionary Views" for more information about the command rule-related Oracle Data Vault data dictionary views

How Command Rules Work

"How Realms Work" on page 5-10 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 command rule to control whether or not users can create tables in the `SCOTT` schema.

Topics:

See Also: "Tutorial: Creating an Email Alert for Security Violations" on page 6-13 for another example of how a command rule can work with a rule set to send an email alert when a violation occurs

Step 1: Connect as User SCOTT and 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, enter a command similar to the following:

   ```
   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;
   ```

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

   ```
   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: Connect Using the DVOWNER or DV_ADMIN Role and Create a Command Rule

After the table has been created in the SCOTT schema, you can create a command rule.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the DV_OWNER or DV_ADMIN role and the SELECT ANY DICTIONARY system privilege.

   "Logging into Oracle Database Vault" on page 3-7 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.
   - **Evaluating 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 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 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 a set of 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 Appendix B, "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.

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`. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Cloud Control, refer to its online Help. See *Oracle Database Performance Tuning Guide* to learn how to monitor database performance, and *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements.

Related Reports and Data Dictionary View

Table 7–2 lists Oracle Database Vault reports that are useful for analyzing command rules. See Chapter 24, "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" on page 24-5	Lists audit records generated by command rule processing operations
"Command Rule Configuration Issues Report" on page 24-3	Tracks rule violations, in addition to other configuration issues the command rule may have
"Object Privilege Reports" on page 24-6	Lists object privileges that the command rule affects
"Sensitive Objects Reports" on page 24-8	Lists objects that the command rule affects
"Rule Set Configuration Issues Report" on page 24-3	Lists rules sets that have no rules defined or enabled, which may affect the command rules that use them

You can use the `DVSYS.DBA_DV_COMMAND_RULE` data dictionary view to find the SQL statements that are protected by command rules. See "DVSYS.DBA_DV_COMMAND_RULE View" on page 22-4 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.

Topics:

See Also: Chapter 16, "Oracle Database Vault Factor APIs"

What Are Factors?

A **factor** is a named variable or attribute, such as a user location, database IP address, or session user, 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.

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" on page 8-5 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 on page 8-32 for more information.

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.

See Also:

- "Default Factors" on page 8-2
- "Oracle Database Vault DVF PL/SQL Factor Functions" on page 16-24
- "Setting the Factor Identification Information" on page 8-6
- "Integrating Oracle Database Vault with Oracle Label Security" on page 10-4
- "Tutorial: Integrating Oracle Database Vault with Oracle Label Security" on page 10-8
- "Related Reports and Data Dictionary Views" on page 8-32
- Chapter 16, "Oracle Database Vault Factor APIs"

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" on page 16-24 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, 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.

- **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.
- **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/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.
- **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.

Topics:

- Accessing the Create Factors Page
- Completing the General Page for Factor Creation
- Configurations Page for Factor Creation
- Options Page of Factor Creation
- Creating and Configuring a Factor Identity

See Also:

- "Adding an Identity to a Factor" on page 8-12
- "Guidelines for Designing Factors" on page 8-31
- "Propagating Oracle Database Vault Policies to Other Databases" on page 11-1

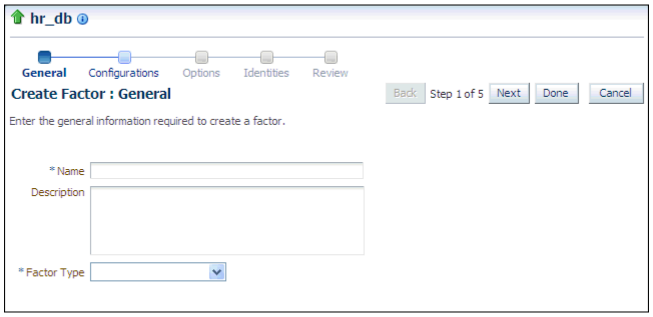
Accessing the Create Factors Page

The Create Factors page enables you to create the factor through a series of pages, starting with a general definition of the factor that you want to create.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

"Logging into Oracle Database Vault" on page 3-7 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.



Create Factor : General

Enter the general information required to create a factor.

* Name	
Description	
* Factor Type	

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" on page 16-24 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 DVSYS.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 enables you to define settings such as the factor's identification, the evaluation method, the Oracle Label Security labeling, the retrieval method, and the validation method for the factor.

Topics:

- "Setting the Factor Identification Information" on page 8-6
- "How Factor Identities Work" on page 8-7
- "Setting the Evaluation Information for a Factor" on page 8-8
- "Setting the Oracle Label Security Labeling Information for a Factor" on page 8-8
- "Setting the Retrieval Method for a Factor" on page 8-8
- "How Retrieval Methods Work" on page 8-8
- "Setting the Validation Method for a Factor" on page 8-9

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 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, 2014, the **By Method** option would return the following value:

    ```
    2014-12-15
    ```

  - **By Constant:** Resolves the factor identity by retrieving the constant value found in the **Retrieval Method** field.
  - **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" on page 8-15 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. See "Related Reports and Data Dictionary Views" on page 8-32 for more information.

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 `DVSYS.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
```

See Also: "Adding an Identity to a Factor" on page 8-12 for more information about factor 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" on page 8-32 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. (See also "Integrating Oracle Database Vault with Oracle Label Security" on page 10-4 for information on integrating OLS labels with a factors.

- 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.

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" on page 16-24

- "DBMS_MACADM Factor Procedures and Functions" on page 16-1
- Chapter 19, "Oracle Database Vault Utility APIs"

See also the default factors provided with Oracle Database Vault for examples of retrieval methods. "Default Factors" on page 8-2 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 `GRANT 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

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 `DVSYS.GET_FACTOR` function) or the value to be assigned to a factor (with the `DVSYS.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 `GRANT 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.

See Also: See the following sections for factor-specific and general utility functions that you can use to build the validation method:

- "DBMS_MACADM Factor Procedures and Functions" on page 16-1
- "Oracle Database Vault Run-Time PL/SQL Procedures and Functions" on page 16-20
- "Oracle Database Vault DVF PL/SQL Factor Functions" on page 16-24
- Chapter 19, "Oracle Database Vault Utility APIs"

Options Page of Factor Creation

The Options page enables you to assign a rule set to a factor, set error options, and if you are not using unified auditing, set audit options.

Topics:

- Assigning a Rule Set to a Factor
- Setting Error Options for a Factor
- Setting Audit Options for a Factor
- How Factor Auditing Works

Assigning a Rule Set to a Factor

Under Assignment Rule Set, you can select a rule set if you want to use a rule set to control when and how a factor identity is set.

For example, you can use a rule set to determine when a database session originates from a known application server or program. Chapter 6, "Configuring Rule Sets," explains how to create rule sets.

- In the Options page, under Assigning a rule set, 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 `DVSYS.SET_FACTOR`, for example:

```
BEGIN
 DVSYS.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. See "How Factors Are Set" on page 8-19 for more information.

Setting Error Options for a Factor

Under Error Options, you must set the processing that must occur when a factor identity cannot be resolved.This attribute is mandatory.

- 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 select from the settings to generate an audit trail if you are not using a unified audit environment. This setting is mandatory.

- 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.
  - **Sometimes:** Creates an audit record based on one or more conditions. When you select **Sometimes**, by default the **Retrieval Error** and **Retrieval NULL** options are selected.

    You can select from the following conditions listed next.

  Conditions that you can select for the **Always** and **Sometimes** options are as follows:

  - **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`).
  - **Retrieval NULL:** Creates an audit record when the identity of a factor is resolved to `NULL`.
  - **Validation Error:** Creates an audit record when the validation method (if provided) returns an error.
  - **Validation False:** Creates an audit record when the validation method (if provided) returns `FALSE`.
  - **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" on page 8-13 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.

How Factor Auditing Works

In a non-unified auditing environment, Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` table, described in Appendix A, "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 "Related Reports and Data Dictionary Views" on page 8-32 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.

Topics:

- About Factor Identities
- About Trust Levels
- About Label Identities
- Creating and Configuring a Factor Identity
- Deleting a Factor Identity
- Using Identity Mapping to Configure an Identity to Use Other Factors

About Factor Identities

An identity is the actual value of the factor. For example, the identity of an IP_Address factor could be the IP address of 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" on page 8-4. 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

See Also:

- "How Factors Work" on page 8-17 for more information about how a factor behaves during a database session
- "Tutorial: Restricting User Activities Based on Session Data" on page 8-25 for an example of how to create and use factor identities

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 `DVSYS.GET_TRUST_LEVEL` to find trust levels greater than 5:

```
DVSYS.GET_TRUST_LEVEL('Network') > 5
```

Or, you can use a `SELECT` statement on the `DVSYS.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 DVSYS.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.

See Chapter 13, "Oracle Database Vault Realm APIs," for more information about the Oracle Database Vault factor-related functions.

About Label Identities

You can assign 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. For more information about labels, see *Oracle Label Security Administrator's Guide*. 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.

Creating and Configuring a Factor Identity

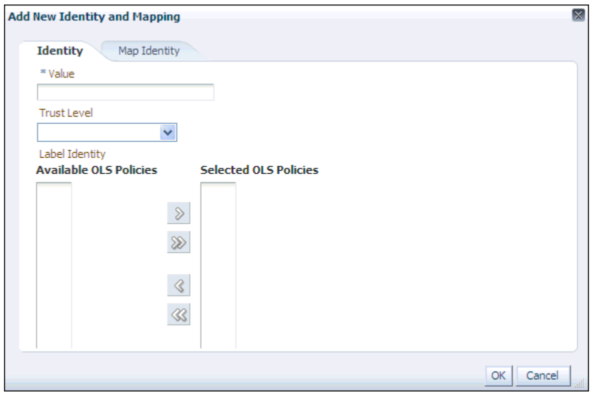
You can create and configure a factor identity in Oracle Database Vault Administrator.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 explains how to log in.
2. In the Administration page, under Database Vault Components, click **Factors**.
3. From the list, select the factor with which you want to create an identity, and then click **Edit**.

   The General page appears.
4. Click **Next** until you reach the Identities page.
5. Select **Add New Identity.**

The Add New Identity and Mapping window appears.



6. 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" on page 8-12 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" on page 8-13 for detailed information about label identities.
7. Click **OK** to return to the Create Factors : Identities page.
8. Click **Done**, and then click **Finish**.

Deleting a Factor Identity

If you want to delete a factor identity, you can locate the various references to it by querying the factor-related Oracle Database Vault views.

See Chapter 22, "Oracle Database Vault Data Dictionary Views," for more information.

1. In the Edit Factor page, click Next until you reach the Identities page.
2. Select the factor identity that you want to remove.
3. Click **Remove**.
4. Click **Done**, then click **Finish**.

Using Identity Mapping to Configure an Identity to Use Other Factors

You can use identity mapping as a way of using a group of factors to manage a set of discrete identity values.

Topics:

- About Identity Mapping
- Mapping an Identity to a Factor

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).

See Also:

- "Identity Configuration Issues Report" on page 24-4
- "Tutorial: Restricting User Activities Based on Session Data" on page 8-25 for an example of how to use identity mapping

Mapping an Identity to a Factor

You can map an identity to a factor by creating a parent-child relationship with two factors.

1. Create a parent factor and set the attribute **Factor Identification** to **By Factors**.

   "Creating a Factor" on page 8-4 describes how to create factors.

2. In the Identities page, for the parent factor, create a new factor identity.

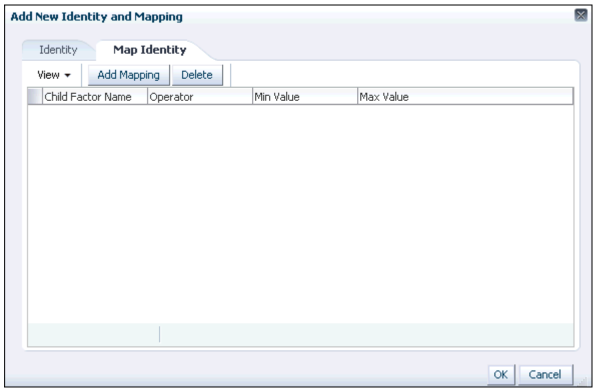
   "Creating and Configuring a Factor Identity" on page 8-13 describes how to create an identity.

3. Map the factor-identity pair of the parent to the factor-identity pairs of its children. Use the following process:

   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 Add New Identity and Mapping window, ensure that at least the **Value** field is filled out in the Identity subpage.

   c. Click the **Map Identity** subpage.



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 can locate the various references to the factor and its identities by querying the factor-related Oracle Database Vault views.

See Chapter 22, "Oracle Database Vault Data Dictionary Views," for more information.

1. Delete any references to the factor, such as factor identities and Oracle Label Security policy associations.

   You cannot delete a factor that has references.

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
- How Factors Are Retrieved
- How Factors Are Set

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" on page 8-24 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 `DVSYS.GET_FACTOR` function.

To find a listing of available factors, query the `DVS.DBA_DV_FACTOR` data dictionary view, described in"DVSYS.DBA_DV_FACTOR View" on page 22-6.

Example 8–1 shows an example of using the `DVSYS.GET_FACTOR` function.

Example 8–1 Using DVSYS.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 `DVSYS.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" on page 16-24 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" on page 8-17.

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" on page 8-17, 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 have a factor identity assigned at any time during a database session, but only if you have defined a factor assignment rule set and that rule set evaluates to true.

You can do this in the application code by using the `DVSYS.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 DVSYS.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" on page 8-17 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 `DVSYS.SET_FACTOR` function.

Tutorial: Preventing Ad Hoc Tool Access to the Database

This tutorial demonstrates how you can use factors to prevent ad hoc tools (such as SQL*Plus) from accessing the database.

Topics:

- About This Tutorial
- Step 1: Enable the HR and OE User Accounts
- Step 2: Create the Factor
- Step 3: Create the Rule Set and Rules
- Step 4: Create the CONNECT Command Rule
- Step 5: Test the Ad Hoc Tool Access Restriction
- Step 6: Remove the Components for This Tutorial

See Also:

- "Tutorial: Restricting User Activities Based on Session Data" on page 8-25 for an example of using factor identity mapping
- "Tutorial: Integrating Oracle Database Vault with Oracle Label Security" on page 10-8 for an example of integrating an Oracle Database Vault factor with an Oracle Label Security label

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, so ensure that this account is active.

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
   ```

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

   For example:

   ```
   CONNECT 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.

3. Check the status of the `HR` account.

   ```
   SELECT USERNAME, ACCOUNT_STATUS FROM DBA_USERS WHERE USERNAME = 'HR';
   ```

4. 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;
   ```

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

5. 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. It 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 have been logged in to SQL*Plus all along, but note that you do not need to restart your SQL*Plus session in order for the Oracle Database Vault changes to take effect. They take effect right away.

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 dbv_owner --Or, CONNECT dbv_owner@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 you can restrict user activities based on their session data, such as the domain the user is using.

Topics:

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;
   ```

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

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

   For example:

   ```
   CONNECT 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.

3. 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

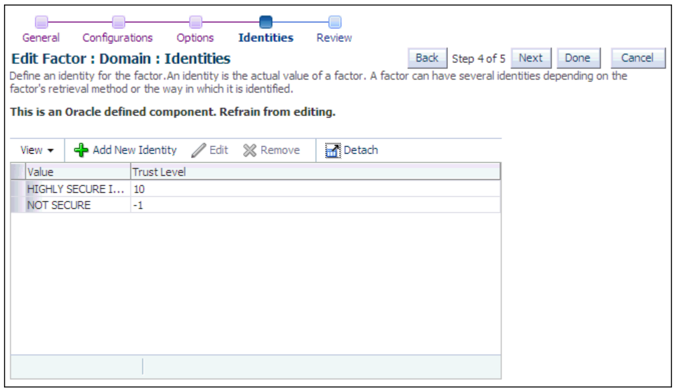
Next, you must add identities to the Domain factor, which is a default factor.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` system privilege.

   "Logging into Oracle Database Vault" on page 3-7 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**.



Edit Factor : Domain : Identities

Define an identity for the factor.An identity is the actual value of a factor. A factor can have several identities depending on the factor's retrieval method or the way in which it is identified.

This is an Oracle defined component. Refrain from editing.

Value	Trust Level
HIGHLY SECURE I...	10
NOT SECURE	-1

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, which 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 Rules Associated with Rule Sets 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 **Add Existing Rule** pages 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" on page 10-4 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" on page 8-2 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`. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Cloud Control, refer to its online Help. See *Oracle Database Performance Tuning Guide* to learn how to monitor database performance, and *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements.

Related Reports and Data Dictionary Views

Table 8–1 lists Oracle Database Vault reports that are useful for analyzing factors and their identities. See Chapter 24, "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" on page 24-5	Audits factors (for example, to find factors that failed to be evaluated)
"Factor Configuration Issues Report" on page 24-4	Lists configuration issues, such as disabled or incomplete rule sets, or to audit issues that may affect the factor
"Factor Without Identities Report" on page 24-4	Lists factors that have had no identities assigned yet
"Identity Configuration Issues Report" on page 24-4	Lists factors that have invalid label identities or no map for the identity
"Rule Set Configuration Issues Report" on page 24-3	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
"DVSYS.DBA_DV_FACTOR View" on page 22-6	Lists the existing factors in the current database instance
"DVSYS.DBA_DV_FACTOR_LINK View" on page 22-8	Shows the relationships of each factor whose identity is determined by the association of child factors
"DVSYS.DBA_DV_FACTOR_TYPE View" on page 22-8	Lists the names and descriptions of factor types used in the system
"DVSYS.DBA_DV_IDENTITY View" on page 22-9	Lists the identities for each factor
"DVSYS.DBA_DV_IDENTITY_MAP View" on page 22-9	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.

Topics:

See Also: Chapter 17, "Oracle Database Vault Secure Application Role APIs"

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.

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.

See Also:

- "Related Reports and Data Dictionary View" on page 9-9 for information about how you can run reports secure application roles that you created in Oracle Database Vault
- Chapter 17, "Oracle Database Vault Secure Application Role APIs" for information about using the PL/SQL interfaces and packages to create and manage Oracle Database Vault secure application roles

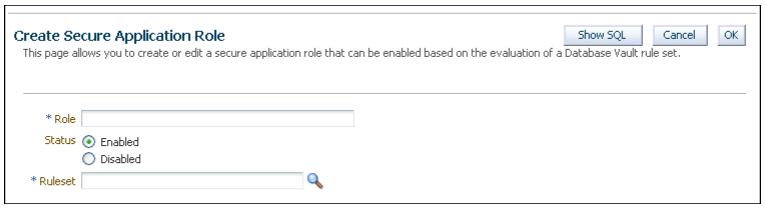
Creating Oracle Database Vault Secure Application Roles

You can create a Database Vault secure application role in Database Vault Administrator.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the DV_OWNER or DV_ADMIN role.

   "Logging into Oracle Database Vault" on page 3-7 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. See Chapter 6, "Configuring Rule Sets," for more information about rule sets.
3. In the Administration page, under Database Vault Components, click **Secure Application Roles**.
4. In the Secure Application Role page:
   - To create a new secure application role, click **Create** to display the Create Secure Application Role page.


Create Secure Application Role

This page allows you to create or edit a secure application role that can be enabled based on the evaluation of a Database Vault rule set.

5. In the Create Role page, enter the following settings under General:
   - **Role:** 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.

       See "SET_ROLE Procedure" on page 17-5 for more information about this function.

– **Disabled:** Disables the role from being available for use. The `DBMS_MACSEC_ROLES.SET_ROLE` function will not be able to enable the role.

6. In the Create Role page, under 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.

   See Chapter 6, "Configuring Rule Sets," for more information about rule sets.

7. Click **OK**.

See Also: "Propagating Oracle Database Vault Policies to Other Databases" on page 11-1

Modifying 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.

See Also: "SET_ROLE Procedure" on page 17-5 for more information about the `SET_ROLE` function

Securing an Oracle Database Vault Secure Application Role

Users who have database administrative privileges may try to use the `DROP ROLE` SQL statement to delete secure application roles that were created using Oracle Database Vault.

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. If necessary, locate the various references to the secure application roles by querying the role-related Oracle Database Vault views.

   See Chapter 22, "Oracle Database Vault Data Dictionary Views," for more information.

2. Check and modify any applications that may be using the secure application role that you want to delete.
3. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 explains how to log in.
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**.

How Oracle Database Vault Secure Application Roles Work

The process flow for a secure application role that is managed by Oracle Database Vault 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" on page 17-1 for more information.
2. Modify your application to call the role, by using the `DBMS_MACSEC_ROLES.SET_ROLE` function.

   See "SET_ROLE Procedure" on page 17-5 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 you can create a secure application role that uses a rule set to control a user's access to the `OE.ORDERS` table during work hours.

Topics:

- About This Tutorial
- Step 1: Create Users for This Tutorial
- Step 2: Enable the OE User Account
- Step 3: Create the Rule Set and Its Rules
- Step 4: Create the Database Vault Secure Application Role
- Step 5: Grant the SELECT Privilege to the Secure Application Role
- Step 6: Test the Database Vault Secure Application Role
- Step 7: Remove the Components for This Tutorial

About This Tutorial

In this tutorial, you restrict the `SELECT` SQL 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
   ```

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

   For example:

   ```
   CONNECT 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.

3. 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;
   ```

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

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. From Cloud Control, log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` system privilege.

   "Logging into Oracle Database Vault" on page 3-7 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 Rules Associated with Rule Sets 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 are ready to 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" on page 8-27, in Chapter 8.)

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. From Cloud Control, log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 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 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`.

For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Cloud Control, refer to its online Help. See *Oracle Database Performance Tuning Guide* to learn how to monitor database performance, and *Oracle Database SQL Tuning Guide* to monitor the performance of individual SQL and PL/SQL statements.

Related Reports and Data Dictionary View

Table 9–1 lists Oracle Database Vault reports that are useful for analyzing Oracle Database Vault secure application roles. See Chapter 24, "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" on page 24-6	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" on page 24-4	Lists secure application roles that have nonexistent database roles, or incomplete or disabled rule sets
"Rule Set Configuration Issues Report" on page 24-3	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" on page 24-11	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 "DVSYS.DBA_DV_ROLE View" on page 22-16 for more information.

10 Integrating Oracle Database Vault with Other Oracle Products

You can integrate Oracle Database Vault with Enterprise User, Transparent Data Encryption (TDE), Oracle Virtual Private Database, Oracle Label Security, and Oracle Data Guard.

Topics:

- Integrating Oracle Database Vault with Enterprise User Security
- Configuring Oracle Database Vault Accounts as Enterprise User Accounts
- Integration of Oracle Database Vault with Transparent Data Encryption
- Attaching Factors to an Oracle Virtual Private Database
- Integrating Oracle Database Vault with Oracle Label Security
- Integrating Oracle Database Vault with Oracle Data Guard

Integrating Oracle Database Vault with Enterprise User Security

You can integrate Oracle Database Vault with Oracle Enterprise User Security.

Topics:

- About Integrating Oracle Database Vault with Enterprise User Security
- Configuring an Enterprise User Authorization

About Integrating Oracle Database Vault with Enterprise User Security

Enterprise User Security enables you to centrally manage 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" on page 6-6 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" on page 6-3 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" on page 5-9 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 dbv_owner -- Or, CONNECT dbv_owner@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.

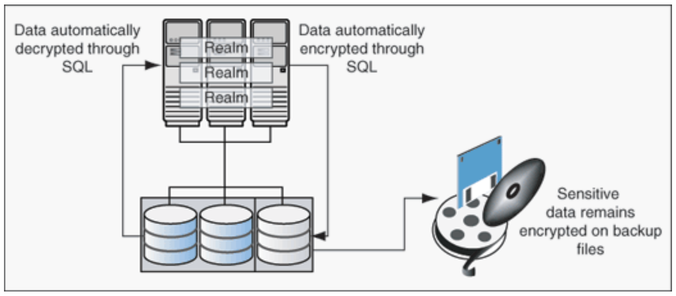
With Transparent Data Encryption, a database administrator or database security administrator can simply encrypt columns with sensitive content in application tables, or encrypt entire application tablespaces, without any modification to the application. For detailed information about Transparent Data Encryption, see *Oracle Database Advanced Security Guide*.

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 10–1 shows how Oracle Database Vault realms handle encrypted data.

Figure 10–1 Encrypted Data and Oracle Database Vault


Data automatically decrypted through SQL
Realm
Realm
Realm
Data automatically encrypted through SQL
Sensitive data remains encrypted on backup files

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 for more information about Oracle Virtual Private Database

Integrating Oracle Database Vault with Oracle Label Security

Oracle Database Vault enables you to integrate Database Vault with Oracle Label Security, and provides reports and data dictionary views to check the integration.

Topics:

- How Oracle Database Vault Is Integrated with Oracle Label Security

- Requirements for Using Oracle Database Vault with Oracle Label Security
- Using Oracle Database Vault Factors with Oracle Label Security Policies
- Tutorial: Integrating Oracle Database Vault with Oracle Label Security
- Related Reports and Data Dictionary Views

See Also: Chapter 18, "Oracle Database Vault Oracle Label Security APIs"

How Oracle Database Vault Is Integrated with Oracle Label Security

When you integrate Oracle Database Vault with Oracle Label Security, it means that you can assign an Oracle Label Security label to an Oracle 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" on page 10-8 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" on page 24-6
- Appendix A, "Auditing Oracle Database Vault" for information about how auditing works for Database Vault
- "Oracle Database Vault Oracle Label Security APIs" on page 18-1 for information about Database Vault APIs that you can use to integrate Database Vault with Oracle Label Security
- "Related Reports and Data Dictionary Views" on page 10-12 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;
  ```

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

Topics:

- About Using Oracle Database Vault Factors with Oracle Label Security Policies

- Configuring Factors to Work with an Oracle Label Security Policy

About Using Oracle Database Vault Factors with Oracle Label Security Policies

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 *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. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Logging into Oracle Database Vault" on page 3-7 explains how to log in.

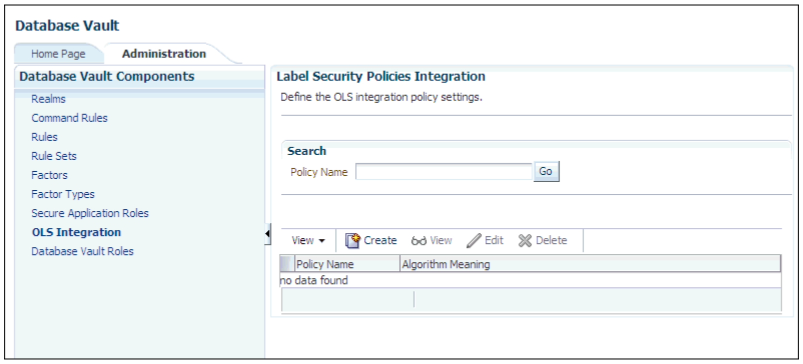
2. Make the user `LBACSYS` account an owner of the realm that contains the schema to which a label security policy has been applied.

   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" on page 5-9 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**.



Define the OLS integration policy settings.

Policy Name	Algorithm Meaning
no data found	

5. In the Label Security Policies 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.

  For more information on these label-merging algorithms, see *Oracle Label Security Administrator's Guide*. If you want to use the `DBMS_MACADM` package to specify a merge algorithm, see Table 18–3, " Oracle Label Security Merge Algorithm Codes" on page 18-2 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" on page 8-12 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

This tutorial shows how you can integrate Oracle Database Vault with Oracle Label Security to grant two administrative users who normally have the same privileges different levels of access.

Topics:

- About This Tutorial
- Step 1: Create Users for This Tutorial
- Step 2: Create the Oracle Label Security Policy
- Step 3: Create Oracle Database Vault Rules to Control the OLS Authorization
- Step 4: Update the ALTER SYSTEM Command Rule to Use the Rule Set
- Step 5: Test the Authorizations
- Step 6: Remove the Components for This Tutorial

About This Tutorial

You can use Oracle Database Vault factors with Oracle Label Security and Oracle Virtual Private Database (VPD) technology to restrict access to sensitive data.

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
   ```

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

   For example:

   ```
   CONNECT 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.

3. 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;
   ```

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

4. 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 are ready to create the Oracle Label Security policy and grants 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 have created the Oracle Label Security policy, you are ready to 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 DVSYS.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

Table 10–1 lists Oracle Database Vault reports that are useful for analyzing the integration of Oracle Database Vault and Oracle Label Security. See Chapter 24, "Oracle Database Vault Reports," for information about how to run these reports.

Table 10–1 Reports Related to Database Vault and Oracle Label Security Integration

Report	Description
"Factor Configuration Issues Report" on page 24-4	Lists factors in which the Oracle Label Security policy does not exist.
"Identity Configuration Issues Report" on page 24-4	Lists invalid label identities (the Oracle Label Security label for this identity has been removed and no longer exists).
"Security Policy Exemption Report" on page 24-12	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 10–2 lists data dictionary views that provide information about existing Oracle Label Security policies used with Oracle Database Vault.

Table 10–2 Data Dictionary Views Used for Oracle Label Security

Data Dictionary View	Description
"DVSYS.DBA_DV_MAC_POLICY View" on page 22-10	Lists the Oracle Label Security policies defined
"DVSYS.DBA_DV_MAC_POLICY_FACTOR View" on page 22-11	Lists the factors that are associated with Oracle Label Security policies
"DVSYS.DBA_DV_POLICY_LABEL View" on page 22-12	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

This section covers the following topics:

- Step 1: Configure the Primary Database
- Step 2: Configure the Standby Database

Step 1: 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

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, query the `DBA_PDBS` data dictionary view. 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 USING CURRENT LOGFILE THROUGH
   ALL SWITCHOVER DISCONNECT;
   ```

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.

11
DBA Operations in an Oracle Database Vault Environment

Database administrators can perform a specific set of operations in an Oracle Database Vault environment, such as using Database Vault with products such as Oracle Data Pump and Oracle GoldenGate.

Topics:

- Using Oracle Database Vault with Oracle Enterprise Manager
- Using Oracle Data Pump with Oracle Database Vault
- Using Oracle Scheduler with Oracle Database Vault
- Oracle Recovery Manager with Oracle Database Vault
- Privileges for Using Oracle Streams with Oracle Database Vault
- Privileges for Using XStream with Oracle Database Vault
- Privileges for Using Oracle GoldenGate in with Oracle Database Vault
- Using Data Masking in an Oracle Database Vault Environment
- Plugging a Database Vault-Enabled PDB to a CDB
- Using the ORADEBUG Utility with Oracle Database Vault

Using Oracle Database Vault with Oracle Enterprise Manager

Database Vault administrators can perform tasks in Oracle Enterprise Manager Cloud Control (Cloud Control) such as propagate polices to other databases and configuring Cloud Control alerts for Database Vault policies.

Topics:

- Propagating Oracle Database Vault Policies to Other Databases
- Enterprise Manager Cloud Control Alerts for Oracle Database Vault Policies
- Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control
- Changing the DBSNMP Account Password in an Oracle Database Vault Environment

Propagating Oracle Database Vault Policies to Other Databases

You can propagate Database Vault policies to other Database Vault-protected databases.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

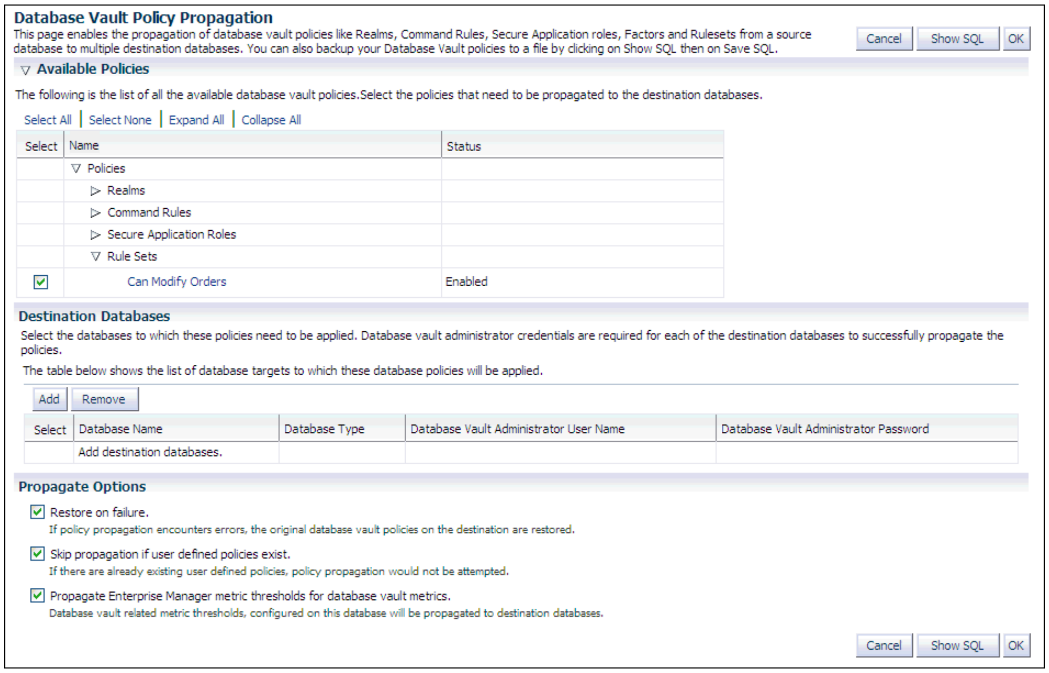
   "Logging into Oracle Database Vault" on page 3-7 explains how to log in.

2. In the Database Vault home page, under Policy Propagation, select **Database Vault Policy Propagation**.

   The Available Policies area in the Policy Propagation subpage lists a summary of the Oracle Database Vault policies that were created for the current database. From here, you can propagate these policies to another database.

3. Under Available Policies, select each policy that you want to propagate to another database.

   By default, all policies are selected.



Database Vault Policy Propagation

This page enables the propagation of database vault policies like Realms, Command Rules, Secure Application roles, Factors and Rulesets from a source database to multiple destination databases. You can also backup your Database Vault policies to a file by clicking on Show SQL then on Save SQL.

Cancel | Show SQL | OK

▽ **Available Policies**

The following is the list of all the available database vault policies.Select the policies that need to be propagated to the destination databases.

Select All | Select None | Expand All | Collapse All

Select	Name	Status
	▽ Policies	
	▷ Realms	
	▷ Command Rules	
	▷ Secure Application Roles	
	▽ Rule Sets	
✓	Can Modify Orders	Enabled

Destination Databases

Select the databases to which these policies need to be applied. Database vault administrator credentials are required for each of the destination databases to successfully propagate the policies.

The table below shows the list of database targets to which these database policies will be applied.

Add | Remove

Select	Database Name	Database Type	Database Vault Administrator User Name	Database Vault Administrator Password
	Add destination databases.			

Propagate Options

✓ Restore on failure.
If policy propagation encounters errors, the original database vault policies on the destination are restored.

✓ Skip propagation if user defined policies exist.
If there are already existing user defined policies, policy propagation would not be attempted.

✓ Propagate Enterprise Manager metric thresholds for database vault metrics.
Database vault related metric thresholds, configured on this database will be propagated to destination databases.

Cancel | Show SQL | OK

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 policies. 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 policies 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 policies.

   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 policy propagation 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 policies, then the policy propagation 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 policies are cleared, and the policies 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 policies 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 policies are active right away in their destination databases.

Enterprise Manager Cloud Control Alerts for Oracle Database Vault Policies

Cloud Control generates Oracle Database Vault-specific alerts. To view these 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 the specific types of 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.

See Also: Chapter 24, "Oracle Database Vault Reports"

Changing the DBSNMP Account Password in an Oracle 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.

Topics:

- About Using Oracle Data Pump with Oracle Database Vault
- Authorizing Users for Oracle Data Pump Regular Export and Import Operations
- Authorizing Users for Oracle Data Pump Transportable Export and Import Operations
- Guidelines for Exporting or Importing Data in an Oracle Database Vault Environment

About Using Oracle Data Pump with Oracle Database Vault

Database administrators who want to use Oracle Data Pump must have Oracle Database Vault-specific authorization, in addition to the standard Oracle Data Pump privileges, if they want to export and import data in an Oracle Database Vault environment.

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 `DVSYS.DBA_DV_DATAPUMP_AUTH` data dictionary view.

See Also:

- *Oracle Database Utilities* for detailed information about Oracle Data Pump
- *Oracle Database Administrator's Guide* for more information about transportable tablespaces
- "DVSYS.DBA_DV_DATAPUMP_AUTH View" on page 22-4

Authorizing Users for Oracle Data Pump Regular Export and Import Operations

You can grant a range of different authorization types for administrators who are responsible for performing regular export and import operations in a Database Vault environment.

Topics:

- About Authorizing Users for Oracle Data Pump Regular Operations
- Levels of Database Vault Authorization for Oracle Data Pump Regular Operations
- Authorizing Users for Oracle Data Pump Regular Operations in Database Vault
- Revoking Oracle Data Pump Authorization from Users

About Authorizing Users for Oracle Data Pump Regular Operations

The Oracle Data Pump authorization that you grant to users enables them to perform regular Oracle Data Pump operations in a Database Vault environment, while full level Data Pump authorization enables them to perform transportable export and import operations as well.

If you want the user only to perform transportable export and import operations, then see "Authorizing Users for Oracle Data Pump Transportable Export and Import Operations" on page 11-7.

Levels of Database Vault Authorization for Oracle Data Pump Regular Operations

Table 11–1 describes the levels of authorization required for Oracle Data Pump regular operations in a Database Vault environment.

Table 11–1 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 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 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.

Table 11–1 (Cont.) Levels of Authorization for Oracle Data Pump Regular Operations

Scenario	**Authorization Required**
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 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 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 for Oracle Data Pump Regular Operations in Database Vault

You can authorize a database administrator 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 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 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 the Data Pump user DP_MGR to export and import objects for the entire database, enter the following:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR');
   ```

   See "AUTHORIZE_DATAPUMP_USER Procedure" on page 20-3 for detailed information about this procedure.

   After you run the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure, you can check the user's authorization by querying the DBA_DV_DATAPUMP_AUTH data dictionary view, described in "DVSYS.DBA_DV_DATAPUMP_AUTH View" on page 22-4.

4. If the user must export the entire database, then grant the user the DV_OWNER role.

```
GRANT DV_OWNER TO DP_MGR;
```

See Also:

- "AUTHORIZE_DATAPUMP_USER Procedure" on page 20-3
- "DVSYS.DBA_DV_DATAPUMP_AUTH View" on page 22-4

Revoking Oracle Data Pump Authorization from Users

You can revoke authorization from the database administrator who is using Oracle Data Pump for regular operations.

1. If you granted the user the `DV_OWNER` role, then optionally revoke this role.

   ```
   REVOKE DV_OWNER FROM DP_MGR;
   ```

2. Query the `DVSYS.DBA_DV_DATAPUMP_AUTH` data dictionary view to find the users who have been granted Oracle Data Pump authorizations.

   ```
   SELECT GRANTEE, SCHEMA, OBJECT FROM DVSYS.DBA_DV_DATAPUMP_AUTH;
   ```

3. Use the information you gathered from Step 2 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');
   ```

   See "UNAUTHORIZE_DATAPUMP_USER Procedure" on page 20-6 for more information.

See Also:

- "UNAUTHORIZE_DATAPUMP_USER Procedure" on page 20-6
- "DVSYS.DBA_DV_DATAPUMP_AUTH View" on page 22-4

Authorizing Users for Oracle Data Pump Transportable Export and Import Operations

You can grant a variety of authorization levels for users who are responsible for using Oracle Data Pump transportable operations.

Topics:

- About Authorizing Users for Oracle Data Pump Transportable Operations
- Levels of Database Vault Authorization for Data Pump Transportable Operations
- Authorizing Users for Data Pump Transportable Operations in Database Vault
- Revoking Transportable Tablespace Authorization from Users

About Authorizing Users for Oracle Data Pump Transportable Operations

If you want users to only have the authorization to perform transportable export and import operations, then you must grant users the correct authorization, based on their tasks.

If your users must have Oracle Data Pump authorization to perform regular operations in a Database Vault environment, then see "Authorizing Users for Oracle Data Pump Regular Export and Import Operations" on page 11-5.

Levels of Database Vault Authorization for Data Pump Transportable Operations

Table 11–2 describes the levels of authorization required for users who want to perform export and import transportable operations in a Database Vault environment.

Table 11–2 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 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 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 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.
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 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 specified in the network link the `DV_DATAPUMP_NETWORK_LINK` role.

Table 11–2 (Cont.) Levels of Authorization for Oracle Data Pump Transporatable Operations

Scenario	Authorization Required
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 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 who 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 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 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 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 who is specified in the network link the `DV_DATAPUMP_NETWORK_LINK` role.

Authorizing Users for Data Pump Transportable Operations in Database Vault

You can authorize users 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 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 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 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 the `DV_DATAPUMP_NETWORK_LINK` role.

   For example:

   ```
   GRANT DV_DATAPUMP_NETWORK_LINK TO DP_MGR;
   ```

6. If the user wants to transportable export or use a network link to transportable import the entire database, then grant this user the `DV_OWNER` role.

   ```
   GRANT DV_OWNER TO DP_MGR;
   ```

   See Also:

   - "DVSYS.DBA_DV_TTS_AUTH View" on page 22-19
   - "DVSYS.DBA_DV_DATAPUMP_AUTH View" on page 22-4
   - "AUTHORIZE_TTS_USER Procedure" on page 20-6
   - "AUTHORIZE_DATAPUMP_USER Procedure" on page 20-3
   - "DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role" on page 12-11

Revoking Transportable Tablespace Authorization from Users

You can revoke authorization from the database administrator who is using Data Pump.

1. If you granted the user the `DV_OWNER` role, then optionally revoke this role.

   ```
   REVOKE DV_OWNER FROM DP_MGR;
   ```

2. Query the `DVSYS.DBA_DV_TTS_AUTH` data dictionary view to find the users who have been granted Oracle Data Pump authorizations.

   ```
   SELECT GRANTEE, TSNAME FROM DVSYS.DBA_DV_TTS_AUTH;
   ```

3. Use the information you gathered from Step 2 to build the `DBMS_MACADM.UNAUTHORIZE_TTS_USER` statement.

   For example:

   ```
   EXEC DBMS_MACADM.UNUTHORIZE_TTS_USER('DP_MGR', 'HR_TS');
   ```

   See "UNAUTHORIZE_TTS_USER Procedure" on page 20-9 for more information.

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.

   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 this user.

   For example:

   ```
   REVOKE DV_DATAPUMP_NETWORK_LINK FROM DP_MGR;
   ```

See Also:

- "DVSYS.DBA_DV_TTS_AUTH View" on page 22-19
- "UNAUTHORIZE_TTS_USER Procedure" on page 20-9
- "UNAUTHORIZE_DATAPUMP_USER Procedure" on page 20-6
- "DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role" on page 12-11

Guidelines for Exporting or Importing Data in an Oracle Database Vault Environment

After you have granted the database administrator who is using Oracle Data Pump the proper authorization, this user is ready to 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 has been granted the `DV_OWNER` role. For detailed information about Oracle Data Pump, see *Oracle Database Utilities*.

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 who 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 want to 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.

Using Oracle Scheduler with Oracle Database Vault

Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific administration, in addition to the standard system privileges required for scheduling database jobs.

Topics:

- About Using Oracle Scheduler with Oracle Database Vault
- Granting a Job Scheduling Administrator Authorization for Oracle Database Vault
- Revoking Authorization from Job Scheduling Administrators

About Using Oracle Scheduler with Oracle Database Vault

The level of authorization that you must grant depends on 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. See "About Realm Authorization" on page 5-9 for instructions on granting a user realm authorization.
- **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.

Granting a Job Scheduling Administrator Authorization for Oracle 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');
   ```

   See "AUTHORIZE_SCHEDULER_USER Procedure" on page 20-5 for detailed information about this procedure.

4. Ensure that the user has been authorized by querying the `DVSYS.DBA_DV_JOB_AUTH` data dictionary view as follows:

   ```
   SELECT GRANTEE,SCHEMA FROM DVSYS.DBA_DV_JOB_AUTH WHERE GRANTEE = 'user_name';
   ```

   See "DVSYS.DBA_DV_JOB_AUTH View" on page 22-10 for more information about this view.

Revoking Authorization from Job Scheduling Administrators

You can revoke authorization from a user for scheduling database jobs.

1. Query the `DVSYS.DBA_DV_JOB_AUTH` data dictionary view to find the user's authorization.

   ```
   SELECT GRANTEE, SCHEMA FROM DVSYS.DBA_DV_JOB_AUTH WHERE GRANTEE='username';
   ```

2. Use the information you gathered from Step 1 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');
   ```

   See "UNAUTHORIZE_SCHEDULER_USER Procedure" on page 20-8 for more information.

Oracle Recovery Manager with 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');
```

See Also:

- "DV_STREAMS_ADMIN Oracle Streams Configuration Role" on page 12-12 for more information about the `DV_STREAMS_ADMIN` role
- "ADD_AUTH_TO_REALM Procedure" on page 13-2 for more information about the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure

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');
```

See Also:

- "DV_XSTREAM_ADMIN XStream Administrative Role" on page 12-13 for more information about the `DV_XSTREAM_ADMIN` role
- "ADD_AUTH_TO_REALM Procedure" on page 13-2 for more information about the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure

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:

- You must be granted the `DV_GOLDENGATE_ADMIN` role in order to configure the Oracle GoldenGate.
- You must be granted the `DV_GOLDENGATE_REDO_ACCESS` role if you want to use the Oracle GoldenGate `TRANLOGOPTIONS DBLOGREADER` method to access redo logs.
- 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');
```

See Also:

- "DV_GOLDENGATE_ADMIN Oracle GoldenGate Administrative Role" on page 12-14 for more information about the `DV_GOLDENGATE_ADMIN` role
- "DV_GOLDENGATE_REDO_ACCESS Oracle GoldenGate Redo Log Access Role" on page 12-14 for more information about the DV_GOLDENGATE_REDO_ACCESS role
- "ADD_AUTH_TO_REALM Procedure" on page 13-2 for more information about the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure

Using Data Masking in an Oracle Database Vault Environment

You can perform data masking in an Oracle Database Vault environment if you have the correct authorizations the database objects that are being masked.

Topics:

- About Data Masking in an Oracle Database Vault Enabled Database
- Adding Data Masking Users to the Data Dictionary Realm Authorizations
- Giving Users Access to Tables or Schemas That They Want to Mask
- Creating a Command Rule to Control Data Masking Privileges

See Also: *Oracle Database Testing Guide* for more information about data masking

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" on page 5-4 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

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 this type of command rule:

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;
   ```

```
/
```

Plugging a Database Vault-Enabled PDB to a CDB

You can plug a legacy Database Vault-enabled PDB to a multitenant container database (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;
   ```

   See *Oracle Database Administrator's Guide* for more information about how to use the `CREATE PLUGGABLE DATABASE` statement.

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 IMMEDIATE
    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 DVSYS.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;
     ```

     See Also:

     - "DVSYS.DBA_DV_ORADEBUG View" on page 22-11
     - "DISABLE_ORADEBUG Procedure" on page 20-11
     - "ENABLE_ORADEBUG Procedure" on page 20-13

12 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.

Topics:

- Oracle Database Vault Schemas
- Oracle Database Vault Roles
- Oracle Database Vault Accounts

Oracle Database Vault Schemas

The Oracle Database Vault objects include two schemas with database tables, sequences, views, triggers, roles, packages, procedures, functions, and contexts that support the administration and run-time processing of Oracle Database Vault.

Oracle Database Vault has the following schemas:

- DVSYS Schema: Owns the Oracle Database Vault schema and related objects
- DVF Schema: Owns the Oracle Database Vault functions that are created to retrieve factor identities

DVSYS Schema

The `DVSYS` schema contains Oracle Database Vault database objects, which 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" on page 12-2 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, which 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 a set of default roles that are based on specific tasks that users must perform, and adhere to separation of duty concepts.

Topics:

- About Oracle Database Vault Roles

- Privileges of Oracle Database Vault Roles
- DV_OWNER Database Vault Owner Role
- DV_ADMIN Database Vault Configuration Administrator Role
- DV_MONITOR Database Vault Monitoring Role
- DV_SECANALYST Database Vault Security Analyst Role
- DV_AUDIT_CLEANUP Audit Trail Cleanup Role
- DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role
- DV_STREAMS_ADMIN Oracle Streams Configuration Role
- DV_XSTREAM_ADMIN XStream Administrative Role
- DV_GOLDENGATE_ADMIN Oracle GoldenGate Administrative Role
- DV_GOLDENGATE_REDO_ACCESS Oracle GoldenGate Redo Log Access Role
- DV_PATCH_ADMIN Database Vault Database Patch Role
- DV_ACCTMGR Database Vault Account Manager Role
- DV_REALM_OWNER Database Vault Realm DBA Role
- DV_REALM_RESOURCE Database Vault Application Resource Owner Role
- DV_PUBLIC Database Vault PUBLIC Role

See Also:

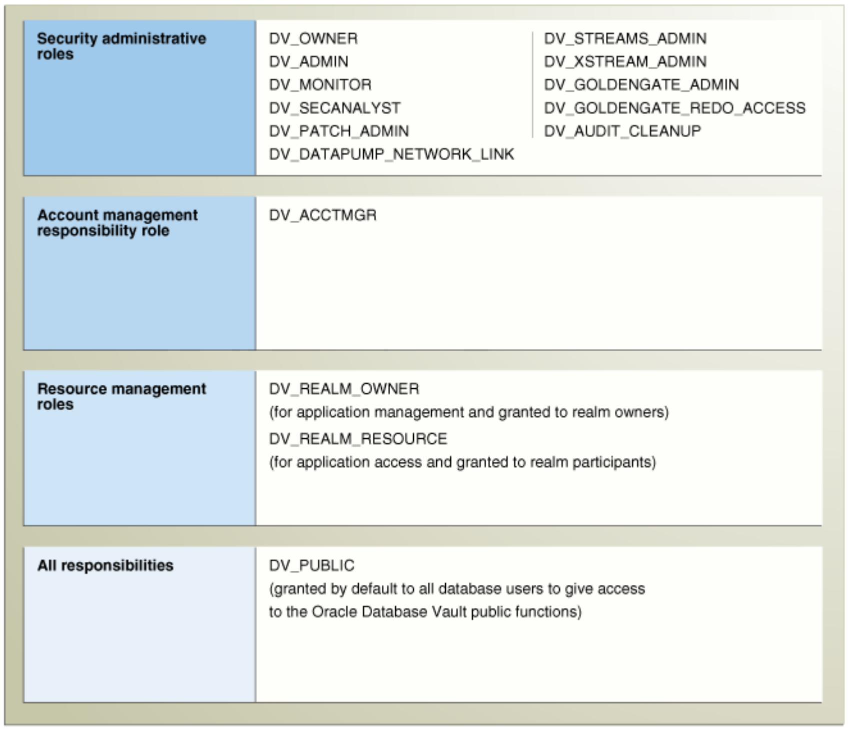
- "Separation of Duty Guidelines" on page D-1
- "Managing Oracle Database Administrative Accounts" on page D-4

About Oracle Database Vault Roles

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

Figure 12–1 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 12–1 How Oracle Database Vault Roles Are Categorized



Security administrative roles	DV_OWNER DV_ADMIN DV_MONITOR DV_SECANALYST DV_PATCH_ADMIN DV_DATAPUMP_NETWORK_LINK	DV_STREAMS_ADMIN DV_XSTREAM_ADMIN DV_GOLDENGATE_ADMIN DV_GOLDENGATE_REDO_ACCESS DV_AUDIT_CLEANUP
Account management responsibility role	DV_ACCTMGR	
Resource management roles	DV_REALM_OWNER (for application management and granted to realm owners) DV_REALM_RESOURCE (for application access and granted to realm participants)	
All responsibilities	DV_PUBLIC (granted by default to all database users to give access to the Oracle Database Vault public functions)	

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: *Oracle Database Security Guide* for general guidelines on managing roles

Privileges of Oracle Database Vault Roles

The `DV_PATCH_ADMIN`, `DV_STREAMS_ADMIN`, `DV_XSTREAM`, `DV_GOLDENGATE_ADMIN`, and `DV_GOLDENGATE_REDO_ACCESS` roles are not included because they have no system privileges.

Table 12–1 summarizes the privileges available with Oracle Database Vault roles.

Table 12–1 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_ AUDIT_ CLEANUP
`DVSYS` schema, `EXEC`	Yes[1]	Yes[2]	No	No	No	No	No	No	No
`DVSYS` packages, `EXECUTE`	Yes	Yes	No	No	No	No	No	No	No
`DVSYS` schema, `SELECT`	Yes	Yes	Yes	Yes, on some Database Vault views[3]	No	No	No	No[4]	Yes, on some Database Vault tables and views[5]
`DVSYS` schema, `DELETE`	No	No	No	No	No	No	No	No	Yes, on some Database Vault tables and view[6]
`DVSYS` schema, grant privileges on objects	No	No	No	No	No	No	No	No	No
`DVF` schema, `EXECUTE`	Yes	No	No	No	No	No	No	No	No
`DVF` schema, `SELECT`	No	No	No	Yes	No	No	No	No	No
Monitor Database Vault	Yes	Yes	Yes	Yes	No	No	No	No	No
Run Database Vault reports	Yes	Yes	No	Yes	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
`SYSMAN` schema, `SELECT`	No	No	No	Yes, portions of	No	No	No	No	No

Table 12–1 (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_ AUDIT_ CLEANUP
`CREATE`, `ALTER`, `DROP` user accounts and profiles[7]	No	No	No	No	Yes	No	No	No	No
Manage objects in schemas that define a realm[8]	No	No	No	No	No	Yes[9]	No	No	No
`RESOURCE` role privileges[10]	No	No	No	No	No	No	Yes	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] `DV_AUDIT_CLEANUP` can perform `SELECT` statements on the `AUDIT_TRAIL$` table and the `DV$ENFORCEMENT_AUDIT`, and `DV$CONFIGURATION_AUDIT` views.

[6] `DV_AUDIT_CLEANUP` can perform `DELETE` statements on the `AUDIT_TRAIL$` table and the `DV$ENFORCEMENT_AUDIT`, and `DV$CONFIGURATION_AUDIT` views.

[7] This privilege does not include the ability to drop or alter the `DVSYS` account, nor change the `DVSYS` password.

[8] This privilege includes `ANY` privileges, such as `CREATE ANY`, `ALTER ANY`, and `DROP ANY`.

[9] The user with this role also must be the realm participant or owner to exercise his or her system privileges.

[10] The `RESOURCE` role provides the following system privileges: `CREATE CLUSTER`, `CREATE INDEXTYPE`, `CREATE OPERATOR`, `CREATE PROCEDURE`, `CREATE SEQUENCE`, `CREATE TABLE`, `CREATE TRIGGER`, `CREATE TYPE`.

DV_OWNER Database Vault Owner Role

The `DV_OWNER` role is the most powerful Database Vault role.

About the DV_OWNER Role

Use the `DV_OWNER` role 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. In addition to `DV_ADMIN` role, the `DV_OWNER` role has the `GRANT ANY ROLE`, `ADMINISTER DATABASE TRIGGER`, and `ALTER ANY TRIGGER` privileges.

To find the full list of system and object privileges associated with the `DV_OWNER` role, you can log into the database instance 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 run 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:** Consider creating a 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.

> **See Also:** Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

DV_ADMIN Database Vault Configuration Administrator Role

The DV_ADMIN role is the second most powerful role, after DV_OWNER.

About the DV_ADMIN 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.

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.

See Also: Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

DV_MONITOR Database Vault Monitoring Role

The DV_MONITOR role is used for monitoring Oracle Database Vault.

About the DV_MONITOR Role

Note: This feature has been updated in Oracle Database 12*c* Release 1 (12.1.0.2).

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.

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.

See Also:

- Chapter 23, "Monitoring Oracle Database Vault"
- Chapter A, "Auditing Oracle Database Vault"
- "Disabling and Enabling Oracle Database Vault" on page B-1

DV_SECANALYST Database Vault Security Analyst Role

The DV_SECANALYST role enables users to analyze activities.

About the DV_SECANALYST Role

Note: This feature has been updated with Oracle Database 12*c* Release 1 (12.1.0.2).

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 Chapter 22, "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.

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.

See Also: Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

DV_AUDIT_CLEANUP Audit Trail Cleanup Role

The DV_AUDIT_CLEANUP role is used for purge operations.

About the DV_AUDIT_CLEANUP Role

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" on page A-5 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.

See Also: Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role

The `DV_DATAPUMP_NETWORK_LINK` role is used for Data Pump import operations.

About the DV_DATAPUMP_NETWORK_LINK Role

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.

See Also: "Using Oracle Data Pump with Oracle Database Vault" on page 11-4

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.

See Also: Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

DV_STREAMS_ADMIN Oracle Streams Configuration Role

The DV_STREAMS_ADMIN role is used with Oracle Streams.

About the DV_STREAMS_ADMIN Role

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.

See Also: Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

DV_XSTREAM_ADMIN XStream Administrative Role

The DV_XSTREAM_ADMIN role is used for Oracle XStream.

About the DV_XSTREAM_ADMIN Role

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.

See Also:

- Appendix B, "Disabling and Enabling Oracle Database Vault"
- "Privileges for Using XStream with Oracle Database Vault" on page 11-14

DV_GOLDENGATE_ADMIN Oracle GoldenGate Administrative Role

DV_GOLDENGATE_ADMIN role is used with Oracle GoldenGate.

About the DV_GOLDENGATE_ADMIN Role

Grant the DV_GOLDENGATE_ADMIN role to 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.

See Also:

- Appendix B, "Disabling and Enabling Oracle Database Vault"
- "Privileges for Using Oracle GoldenGate in with Oracle Database Vault" on page 11-15

DV_GOLDENGATE_REDO_ACCESS Oracle GoldenGate Redo Log Access Role

The DV_GOLDENGATE_REDO_ACCESS role is used with Oracle GoldenGate.

About the DV_GOLDENGATE_REDO_ACCESS Role

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.

See Also:

- Appendix B, "Disabling and Enabling Oracle Database Vault"
- "Privileges for Using Oracle GoldenGate in with Oracle Database Vault" on page 11-15

DV_PATCH_ADMIN Database Vault Database Patch Role

The `DV_PATCH_ADMIN` role is used for patching operations.

About the DV_PATCH_ADMIN Role

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.

See Also: *Oracle Database Security Guide* for information about creating unified audit policies

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_OWNER?

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: Appendix B, "Disabling and Enabling Oracle Database Vault," for information about 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.

About the DV_ACCTMGR Role

Note: This feature has been updated in Oracle Database 12*c* Release 1 (12.1.0.2).

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

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.
- Consider creating 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.

See Also: Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

DV_REALM_OWNER Database Vault Realm DBA Role

The DV_REALM_OWNER role is used for realm management.

About the DV_REALM_OWNER Role

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" on page 5-9 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.

See Also: Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

DV_REALM_RESOURCE Database Vault Application Resource Owner Role

The DV_REALM_RESOURCE role is use for the management of realm resources.

About the DV_REALM_RESOURCE Role

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.

> **See Also:** Appendix B, "Disabling and Enabling Oracle Database Vault," for information about disabling and enabling Oracle Database Vault

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

Oracle Database Vault prompts for two accounts during installation: Oracle Database Vault Owner and Oracle Database Vault Account Manager.

You must supply an account name and password for the Oracle Database Vault Owner account during installation. Creating an Oracle Database Vault Account Manager is optional.

The Oracle Database Vault Owner account is granted the DV_OWNER role. This account can manage Oracle Database Vault roles and configuration. (See "DV_OWNER Database Vault Owner Role" on page 12-6 for detailed information about this role.)

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. (See "DV_ACCTMGR Database Vault Account Manager Role" on page 12-16 for detailed information about this role.)

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.

> **Tip:** Oracle recommends that you grant the DV_OWNER and DV_ACCTMGR roles to existing user accounts. However, continue to maintain the original DV_OWNER and DV_ACCTMGR user accounts that you created during installation. This way, for example, if a user who has been granted one of these roles forgets his or her password, then you can log in as the original Database Vault Account Manager user and then recreate the password without having to disable Oracle Database Vault.

Table 12–2 lists the Oracle Database Vault database accounts that are needed in addition to the accounts that you create during installation.

Table 12–2 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" on page 10-6 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 12–3 lists some model database accounts that can act as a guide. (The accounts listed in Table 12–3 serve as a guide to implementing Oracle Database Vault roles. These are not actual accounts that are created during installation.)

Table 12–3 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
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

See Also: "Configuring Oracle Database Vault Accounts as Enterprise User Accounts" on page 10-2 for information about configuring the Database Vault accounts as enterprise user accounts

13 Oracle Database Vault Realm APIs

The DBMS_MACADM PL/SQL package contains a set of realm-specific procedures.

Topics:

About the DBMS_MACADM Realm Procedures

Table 13–1 lists procedures within the DBMS_MACADM package that you can use to configure 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 19–1 on page 19-2 for more information.

See Also:

- Chapter 5, "Configuring Realms," for detailed information about realms
- Chapter 19, "Oracle Database Vault Utility APIs," for a set of general-purpose utility procedures that you can use with the realm procedures

Table 13–1 DBMS_MACADM Realm Configuration 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
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 and its related Database Vault configuration information
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

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.

For detailed information about realm authorization, see "About Realm Authorization" on page 5-9.

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);
```

Parameters

Table 13–2 ADD_AUTH_TO_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DVSYS.DBA_DV_REALM View" on page 22-14.

Table 13–2 (Cont.) ADD_AUTH_TO_REALM Parameters

Parameter	Description
`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 "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15. To find existing secure application roles used in privilege management, query the `DVSYS.DBA_DV_ROLE` view. Both are described in Chapter 22.
`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 `DVSYS.DBA_DV_RULE_SET` view, described in "DVSYS.DBA_DV_RULE_SET_RULE View" on page 22-19.
`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" on page 5-9 for more information on participants and owners.

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;
/
```

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 13–3 ADD_OBJECT_TO_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DVSYS.DBA_DV_REALM View" on page 22-14
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 "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15.
object_name	Object name. (The wildcard % is allowed. See "Object Name" under "About Realm-Secured Objects" on page 5-9 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 DVSYS.DBA_DV_REALM_OBJECT view, described in "DVSYS.DBA_DV_REALM_OBJECT View" on page 22-16.
object_type	Object type, such as TABLE, INDEX, or ROLE. (The wildcard % is allowed. See "Object Types" under "About Realm-Secured Objects" on page 5-9 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.

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);
```

Parameters

Table 13–4 CREATE_REALM Parameters

Parameter	Description
`realm_name`	Realm name, up to 90 characters in mixed-case. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DVSYS.DBA_DV_REALM View" on page 22-14.
`description`	Description of the purpose of the realm, up to 1024 characters in mixed-case.
`enabled`	`DBMS_MACUTL.G_YES` (Yes) enables realm checking; `DBMS_MACUTL.G_NO` (No) disables it. The default is `DBMS_MACUTL.G_YES`.
`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 "Using Mandatory Realms to Restrict User Access to Objects within a Realm" on page 5-2 for more information about mandatory realms.

Example

```
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);
END;
/
```

See Also: Example 19–1, "Creating a Realm Using DBMS_MACUTL Constants" on page 19-4

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);
```

Parameters

Table 13–5 DELETE_AUTH_FROM_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DVSYS.DBA_DV_REALM View" on page 22-14
`grantee`	User or role name. To find the authorization of a particular user or role, query the `DVA_DV_REALM_AUTH` view, described in "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15.

Example

```
BEGIN
 DBMS_MACADM.DELETE_AUTH_FROM_REALM(
  realm_name => 'Performance Statistics Realm',
  grantee    => 'SYS');
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 13–6 DELETE_OBJECT_FROM_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DVSYS.DBA_DV_REALM View" on page 22-14
`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" on page 5-9 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 `DVSYS.DBA_DV_REALM_OBJECT` view, described in "DVSYS.DBA_DV_REALM_OBJECT View" on page 22-16.
`object_type`	Object type, such as `TABLE`, `INDEX`, or `ROLE`. (The wildcard % is allowed. See "Object Types" under "About Realm-Secured Objects" on page 5-9 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 Database Vault configuration information that specifies who is authorized and what objects are protected. It does not delete the actual database objects or users.

To find users who are authorized for the realm, query the `DVSYS.DBA_DV_REALM_AUTH` view. To find the objects that are protected by the realm, query the `DVSYS.DBA_DV_REALM_OBJECT` view. These views are described in Chapter 22, "Oracle Database Vault Data Dictionary Views."

Syntax

```
DBMS_MACADM.DELETE_REALM(
  realm_name IN VARCHAR2);
```

Parameters

Table 13–7 DELETE_REALM Parameter

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DVSYS.DBA_DV_REALM View" on page 22-14

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 (`DVSYS.DBA_DV_REALM_AUTH` view) and what objects are protected (`DVSYS.DBA_DV_REALM_OBJECT` view).

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 `DVSYS.DBA_DV_REALM` view. To find its authorizations, query `DVSYS.DBA_DV_REALM_AUTH`. Both are described under Chapter 22, "Oracle Database Vault Data Dictionary Views."

Syntax

```
DBMS_MACADM.DELETE_REALM_CASCADE(
  realm_name IN VARCHAR2);
```

Parameters

Table 13–8 DELETE_REALM_CASCADE Parameter

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DVSYS.DBA_DV_REALM View" on page 22-14

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 13–9 RENAME_REALM Parameters

Parameter	Description
`realm_name`	Current realm name. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DVSYS.DBA_DV_REALM View" on page 22-14
`new_name`	New realm name, up to 90 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" on page 22-15.

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);
```

Parameters

Table 13–10 UPDATE_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DVSYS.DBA_DV_REALM View" on page 22-14
`description`	Description of the purpose of the realm, up to 1024 characters in mixed-case.
`enabled`	`DBMS_MACUTL.G_YES` (Yes) enables realm checking; `DBMS_MACUTL.G_NO` (No) disables it. The default for `enabled` is the previously set value, which you can find by querying the `DVSYS.DBA_DV_REALM` data dictionary view.

Table 13–10 (Cont.) UPDATE_REALM Parameters

Parameter	Description
`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 `DVSYS.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 "Using Mandatory Realms to Restrict User Access to Objects within a Realm" on page 5-2 for more information about mandatory realms.

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 + G_REALM_AUDIT_SUCCESS);
END,
  realm_type    => 1);
/
```

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);
```

Parameters

Table 13–11 UPDATE_REALM_AUTH Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DVSYS.DBA_DV_REALM View" on page 22-14.
`grantee`	User or role name. To find the available users and roles, 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 "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15. To find existing secure application roles used in privilege management, query the `DVSYS.DBA_DV_ROLE` view, described in "DVSYS.DBA_DV_ROLE View" on page 22-16.
`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 `DVSYS.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 Chapter 22, "Oracle Database Vault Data Dictionary Views."
`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 `DVSYS.DBA_DV_REALM_AUTH` data dictionary view.

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;
/
```

14

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.

Topics:

- DBMS_MACADM Rule Set Procedures
- Oracle Database Vault PL/SQL Rule Set Functions

DBMS_MACADM Rule Set Procedures

The DBMS_MACADM PL/SQL package provides procedures that enable you to manage both rule sets and rules.

About the DBMS_MACADM Rule Set Procedures

Table 14–1 lists procedures within the DBMS_MACADM package that you can use to configure rule sets. Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures.

Table 14–1 DBMS_MACADM Rule Set Configuration Procedures

Procedure	Description
ADD_RULE_TO_RULE_SET Procedure	Adds a rule to a rule set
CREATE_RULE Procedure	Creates a rule
CREATE_RULE_SET Procedure	Creates a rule set
DELETE_RULE Procedure	Deletes a rule
DELETE_RULE_FROM_RULE_SET Procedure	Deletes a rule from a rule set
DELETE_RULE_SET Procedure	Deletes a rule set
RENAME_RULE Procedure	Renames a rule. The name change takes effect everywhere the rule is used.
RENAME_RULE_SET Procedure	Renames a rule set. The name change takes effect everywhere the rule set is used.
UPDATE_RULE Procedure	Updates a rule
UPDATE_RULE_SET Procedure	Updates a rule set

See Also:

- Chapter 6, "Configuring Rule Sets," for detailed information about rule sets
- Chapter 19, "Oracle Database Vault Utility APIs," for a set of general-purpose utility procedures that you can use with the rule set procedures and functions

ADD_RULE_TO_RULE_SET Procedure

The `ADD_RULE_TO_RULE_SET` procedure adds rule to a rule set, and lets you specify whether to have the rule be 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);
```

Parameters

Table 14–2 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 `DVSYS.DBA_DV_RULE_SET` view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.
`rule_name`	Rule to add to the rule set. To find existing rules, query the `DVSYS.DBA_DV_RULE` view, described in "DVSYS.DBA_DV_RULE View" on page 22-17. To find rules that have been associated with rule sets, use `DVSYS.DBA_DV_RULE_SET_RULE`, described in "DVSYS.DBA_DV_RULE View" on page 22-17.
`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` (Yes; default). Enables the rule to be checked during the rule set evaluation. ▪ `DBMS_MACUTL.G_NO` (No). Prevents the rule from being checked during the rule set evaluation. See Table 19–1 on page 19-2 for more information.

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. After you create a rule, you can add it to a rule set.

Syntax

```
DBMS_MACADM.CREATE_RULE(
  rule_name  IN VARCHAR2,
  rule_expr  IN VARCHAR2);
```

Parameters

Table 14–3 CREATE_RULE Parameters

Parameter	Description
`rule_name`	Rule name, up to 90 characters in mixed-case. Spaces are allowed. To find existing rules in the current database instance, query the `DVSYS.DBA_DV_RULE` view, described in "DVSYS.DBA_DV_RULE View" on page 22-17. To find rules that have been associated with rule sets, query `DVSYS.DBA_DV_RULE_SET_RULE`, described in "DVSYS.DBA_DV_RULE_SET_RULE View" on page 22-19.
`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" on page 6-9 for more information on rule expressions.

Examples

The following example shows how to create a rule expression that checks if the current session user is `SYSADM`.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Check UPDATE operations',
  rule_expr  =>'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SYSADM''');
END;
/
```

Note: The following feature is available starting with Oracle Database 12*c* Release 1 (12.1.0.2).

The following 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 FALSE);
```

Parameters

Table 14–4 CREATE_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name, up to 90 characters in mixed-case. Spaces are allowed. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.
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.

Table 14–4 (Cont.) CREATE_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 (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
`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.

Example

```
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;
```

```
/
```

See Also: Example 19–2, "Creating a Rule Set Using DBMS_MACUTL Constants" on page 19-5

DELETE_RULE Procedure

The `DELETE_RULE` procedure deletes a rule.

Syntax

```
DBMS_MACADM.DELETE_RULE(
  rule_name IN VARCHAR2);
```

Parameter

Table 14–5 DELETE_RULE Parameter

Parameter	Description
`rule_name`	Rule name. To find existing rules in the current database instance, query the `DVSYS.DBA_DV_RULE` view, described in "DVSYS.DBA_DV_RULE View" on page 22-17. To find rules that have been associated with rule sets, query `DVSYS.DBA_DV_RULE_SET_RULE`, described in "DVSYS.DBA_DV_RULE_SET_RULE View" on page 22-19.

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 14–6 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 `DVSYS.DBA_DV_RULE_SET` view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.
`rule_name`	Rule to remove from the rule set. To find existing rules in the current database instance, query the `DVSYS.DBA_DV_RULE` view, described in "DVSYS.DBA_DV_RULE View" on page 22-17. To find rules that have been associated with rule sets, query `DVSYS.DBA_DV_RULE_SET_RULE`, described in "DVSYS.DBA_DV_RULE_SET_RULE View" on page 22-19.

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 14–7 DELETE_RULE_SET Parameter

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.

Example

```
EXEC DBMS_MACADM.DELETE_RULE_SET('Limit_DBA_Access');
```

RENAME_RULE Procedure

The RENAME_RULE procedure renames a rule. The name change takes effect everywhere the rule is used.

Syntax

```
DBMS_MACADM.RENAME_RULE(
  rule_name  IN VARCHAR2,
  new_name   IN VARCHAR2);
```

Parameters

Table 14–8 RENAME_RULE Parameters

Parameter	Description
rule_name	Current rule name. To find existing rules in the current database instance, query the DVSYS.DBA_DV_RULE view, described in "DVSYS.DBA_DV_RULE View" on page 22-17. To find rules that have been associated with rule sets, query DVSYS.DBA_DV_RULE_SET_RULE, described in "DVSYS.DBA_DV_RULE_SET_RULE View" on page 22-19.
new_name	New rule name, up to 90 characters in mixed-case.

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. The name change takes effect everywhere the rule set is used.

Syntax

```
DBMS_MACADM.RENAME_RULE_SET(
  rule_set_name IN VARCHAR2,
  new_name      IN VARCHAR2);
```

Parameters

Table 14–9 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 `DVSYS.DBA_DV_RULE_SET` view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.
`new_name`	New rule set name, up to 90 characters in mixed-case. Spaces are allowed.

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 14–10 UPDATE_RULE Parameters

Parameter	Description
`rule_name`	Rule name. To find existing rules in the current database instance, query the `DVSYS.DBA_DV_RULE` view, described in "DVSYS.DBA_DV_RULE View" on page 22-17. To find rules that have been associated with rule sets, query `DVSYS.DBA_DV_RULE_SET_RULE`, described in "DVSYS.DBA_DV_RULE_SET_RULE View" on page 22-19.
`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" on page 6-9 for more information on rule expressions. To find existing rule expressions, query the `DVSYS.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 FALSE);
```

Parameters

Table 14–11 UPDATE_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, query the `DVSYS.DBA_DV_RULE_SET` view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.
`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 `DVSYS.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 `DVSYS.DBA_DV_RULE_SET` data dictionary view.
`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 `DVSYS.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 `DVSYS.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.

Table 14–11 (Cont.) UPDATE_RULE_SET Parameters

Parameter	Description
`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 `DVSYS.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. ■ `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 (default).

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

In addition to the rule set-specific procedures in the `DBMS_MADADM` PL/SQL package, you can use the standalone Oracle Database Vault PL/SQL rule set functions.

About the Oracle Database Vault PL/SQL Rule Set Functions

Oracle Database Vault provides a set of functions that you can use in rule sets to inspect the SQL statement that you want the rule set to protect.

For example, if a rule set protects `SELECT ON HR.EMPLOYEES` under a command rule, then you could use these functions to make more informed decisions in the rule expression.

Table 14–12 lists the default rule functions.

Table 14–12 Installed Oracle Database Vault PL/SQL Rule Set Functions

Rule Set Function	Description
DVSYS.DV_SYSEVENT Function	Returns the system event firing the rule set
DVSYS.DV_LOGIN_USER Function	Returns the login user name
DVSYS.DV_INSTANCE_NUM Function	Returns the database instance number
DVSYS.DV_DATABASE_NAME Function	Returns the database name
DVSYS.DV_DICT_OBJ_TYPE Function	Returns the type of the dictionary object on which the database operation occurred (for example, table, procedure, view)
DVSYS.DV_DICT_OBJ_OWNER Function	Returns the owner of the dictionary object on which the database operation occurred
DVSYS.DV_DICT_OBJ_NAME Function	Returns the name of the dictionary object on which the database operation occurred
DVSYS.DV_SQL_TEXT Function	Returns the first 4000 characters of SQL text of the database statement used in the operation

DVSYS.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

```
DVSYS.DV_SYSEVENT ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get System Event Firing the Maintenance Rule Set',
  rule_expr => 'DVSYS.DV_SYSEVENT = ''CREATE''');
END;
/
```

DVSYS.DV_LOGIN_USER Function

The `DV_LOGIN_USER` function returns the login user name, in `VARCHAR2` data type.

Syntax

```
DVSYS.DV_LOGIN_USER ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
```

```
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check System Login User Name',
  rule_expr => 'DVSYS.DV_LOGIN_USER = ''SEBASTIAN''');
END;
/
```

DVSYS.DV_INSTANCE_NUM Function

The `DV_INSTANCE_NUM` function returns the database instance number, in `NUMBER` data type.

Syntax

```
DVSYS.DV_INSTANCE_NUM ()
RETURN NUMBER;
```

Parameters

None.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database Instance Number',
  rule_expr => 'DVSYS.DV_INSTANCE_NUM BETWEEN 6 AND 9');
END;
/
```

DVSYS.DV_DATABASE_NAME Function

The `DV_DATABASE_NAME` function returns the database name, in `VARCHAR2` data type.

Syntax

```
DVSYS.DV_DATABASE_NAME ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database Name',
  rule_expr => 'DVSYS.DV_DATABASE_NAME = ''ORCL''');
END;
/
```

DVSYS.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, table, procedure, or view). The return type is `VARCHAR2`.

Syntax

```
DVSYS.DV_DICT_OBJ_TYPE ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Dictionary Object Type',
  rule_expr => 'DVSYS.DV_DICT_OBJ_TYPE IN (''TABLE'', ''VIEW'')');
END;
/
```

DVSYS.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

```
DVSYS.DV_DICT_OBJ_OWNER ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Dictionary Object Owner',
  rule_expr => 'DVSYS.DV_DICT_OBJ_OWNER = ''JSMITH''');
END;
/
```

DVSYS.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

```
DVSYS.DV_DICT_OBJ_NAME ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Dictionary Object Name',
  rule_expr => 'DVSYS.DV_DICT_OBJ_NAME = ''SALES''');
END;
/
```

DVSYS.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

```
DVSYS.DV_SQL_TEXT ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check SQL Text',
  rule_expr => 'DVSYS.DV_SQL_TEXT = ''SELECT SALARY FROM HR.EMPLOYEES''');
END;
/
```

15 Oracle Database Vault Command Rule APIs

The DBMS_MACADM PL/SQL package provides a set of command rule procedures.

Topics:

- About Command Rule Procedures within DBMS_MACADM
- CREATE_COMMAND_RULE Procedure
- DELETE_COMMAND_RULE Procedure
- UPDATE_COMMAND_RULE Procedure

About Command Rule Procedures within DBMS_MACADM

Table 15–1 lists procedures within the DBMS_MACADM package that you can use to configure command rules. Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures.

Table 15–1 DBMS_MACADM Command Rule Configuration 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
DELETE_COMMAND_RULE Procedure	Drops a command rule declaration
UPDATE_COMMAND_RULE Procedure	Updates a command rule declaration

See Also:

- Chapter 7, "Configuring Command Rules," for detailed information about realms
- Chapter 19, "Oracle Database Vault Utility APIs," for a set of general-purpose utility procedures that you can use with the command rule procedures

CREATE_COMMAND_RULE Procedure

The 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.

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);
```

Parameters

Table 15–2 CREATE_COMMAND_RULE Parameters

Parameter	Description
`command`	SQL statement to protect. See also the following: ▪ "DVSYS.DBA_DV_COMMAND_RULE View" on page 22-4 for a listing of existing command rules ▪ "SQL Statements That Can Be Protected by Command Rules" on page 7-3 for a listing of available SQL statements that you can use ▪ *Oracle Database SQL Language Reference* for more information about SQL statements
`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 `DVSYS.DBA_DV_RULE_SET` view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.
`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 or Editing a Command Rule" on page 7-4 for more information.
`object_name`	Object name. (The wildcard % is allowed. See "Object Name" in "Creating or Editing a Command Rule" on page 7-4 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`	`DBMS_MACUTL.G_YES` (Yes) enables the command rule; `DBMS_MACUTL.G_NO` (No) disables it. The default is `DBMS_MACUTL.G_YES`.

Example

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command          => 'SELECT',
  rule_set_name    => 'Limit Sector 2 Access',
  object_owner     => 'SYSADM',
  object_name      => 'EMP_DATA',
  enabled          => DBMS_MACUTL.G_YES);
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);
```

Parameters

Table 15–3 DELETE_COMMAND_RULE Parameters

Parameter	Description
`command`	SQL statement the command rule protects. To find available command rules, query the `DVSYS.DBA_DV_COMMAND_RULE` view, described in "DVSYS.DBA_DV_COMMAND_RULE View" on page 22-4
`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*. See also "Object Owner" in "Creating or Editing a Command Rule" on page 7-4 for more information.
`object_name`	Object name. (The wildcard % is allowed. See "Object Name" in "Creating or Editing a Command Rule" on page 7-4 for more information about objects protected by command rules.) To find the available objects, query the `ALL_OBJECTS` view, described in *Oracle Database Reference*.

Example

```
BEGIN
 DBMS_MACADM.DELETE_COMMAND_RULE(
  command      => 'SELECT',
  object_owner => 'SYSADM',
  object_name  => 'EMP_DATA');
END;
/
```

UPDATE_COMMAND_RULE Procedure

The `UPDATE_COMMAND_RULE` procedure updates a command rule declaration.

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);
```

Parameters

Table 15–4 UPDATE_COMMAND_RULE Parameters

Parameter	Description
`command`	SQL statement to protect. See also the following: ■ "SQL Statements That Can Be Protected by Command Rules" on page 7-3 for a listing of available SQL statements that you can use ■ "DVSYS.DBA_DV_COMMAND_RULE View" on page 22-4 for a listing of existing command rules ■ *Oracle Database SQL Language Reference* for more information about SQL statements
`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 `DVSYS.DBA_DV_RULE_SET` view, described in Chapter 22, "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 or Editing a Command Rule" on page 7-4 for more information.
`object_name`	Object name. (The wildcard % is allowed. See "Object Name" in "Creating or Editing a Command Rule" on page 7-4 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`	`DBMS_MACUTL.G_YES` (Yes) enables the command rule; `DBMS_MACUTL.G_NO` (No) disables it. The default for `enabled` is the previously set value, which you can find by querying the `DVSYS.DBA_DV_COMMAND_RULE` data dictionary view.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'SELECT',
  rule_set_name   => 'Limit Sector 2 Access',
  object_owner    => 'SYSADM',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_NO);
END;
/
```

16 Oracle Database Vault Factor APIs

You can use the DBMS_MACADM PL/SQL package, a set of standalone Oracle Database Vault rule procedures and functions, and DVF PL/SQL functions to manage factors.

Topics:

- DBMS_MACADM Factor Procedures and Functions
- Oracle Database Vault Run-Time PL/SQL Procedures and Functions
- Oracle Database Vault DVF PL/SQL Factor Functions

DBMS_MACADM Factor Procedures and Functions

The DBMS_MACADM PL/SQL package provides procedures that enable you to manage factors, factor types, factor identities.

About the DBMS_MACADM Factor Procedures and Functions

Table 16–1 lists procedures and functions within the DBMS_MACADM package that you can use to configure factors. Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures and functions.

Table 16–1 DBMS_MACADM Factor Configuration Procedures

Procedure	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)
DELETE_FACTOR Procedure	Deletes a factor
DELETE_FACTOR_LINK Procedure	Removes a parent-child relationship for two factors

Table 16–1 (Cont.) DBMS_MACADM Factor Configuration Procedures

Procedure	Description
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

See Also:

- Chapter 8, "Configuring Factors," for detailed information about factors
- Chapter 19, "Oracle Database Vault Utility APIs," for a set of general-purpose utility procedures that you can use with the factor procedures and functions

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 16–2 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 `DVSYS.DBA_DV_FACTOR_LINK` view, described in "DVSYS.DBA_DV_FACTOR_LINK View" on page 22-8.
`child_factor_name`	Child factor name.

Table 16–2 (Cont.) ADD_FACTOR_LINK Parameters

Parameter	Description
`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 Chapter 22, "Oracle Database Vault Data Dictionary Views": ■ `DVSYS.DBA_DV_MAC_POLICY`: Lists Oracle Label Security policies defined in the current database instance. ■ `DVSYS.DBA_DV_MAC_POLICY_FACTOR`: Lists the factors that are associated with Oracle Label Security policies for the current database instance. ■ `DVSYS.DBA_DV_POLICY_LABEL`: Lists the Oracle Label Security label for each factor identifier in the `DVSYS.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 16–3 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 `DVSYS.DBA_DV_MAC_POLICY` view, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10. To find factors that are associated with Oracle Label Security policies, query `DVSYS.DBA_DV_MAC_POLICY_FACTOR`, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10.
`factor_name`	Factor name. To find existing factors, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.

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 16–4 CHANGE_IDENTITY_FACTOR Parameters

Parameter	Description
`factor_name`	Current factor name. To find existing factors, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
`value`	Value of the identity to update. To find existing identities for each factor in the current database instance, query the `DVSYS.DBA_DV_IDENTITY` view, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9. To find current identity mappings, query the `DVSYS.DBA_DV_IDENTITY_MAP` view, described in "DVSYS.DBA_DV_IDENTITY_MAP View" on page 22-9.
`new_factor_name`	Name of the factor to associate with the identity, which you can find by querying the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.

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 16–5 CHANGE_IDENTITY_VALUE Parameters

Parameter	Description
`factor_name`	Factor name. To find existing factors, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
`value`	Current value associated with the identity. To find existing identities for each factor in the current database instance, query the `DVSYS.DBA_DV_IDENTITY` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6. To find current identity mappings, query the `DVSYS.DBA_DV_IDENTITY_MAP` view, described in "DVSYS.DBA_DV_IDENTITY_MAP View" on page 22-9.
`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 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.

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 16–6 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$DATABASE_DOMAIN` function, described in "Oracle Database Vault DVF PL/SQL Factor Functions" on page 16-24.
`domain_host`	Oracle Real Application Clusters host name being added to the domain. To find host name of a database, run the `DVF.F$DATABASE_HOSTNAME` function, described in "Oracle Database Vault DVF PL/SQL Factor Functions" on page 16-24.

Table 16–6 (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 `DVSYS.DBA_DV_MAC_POLICY` view, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10.
`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" on page 16-9.

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 16–7 CREATE_FACTOR Parameters

Parameter	Description
`factor_name`	Factor name, up to 28 characters in mixed-case, without spaces. To find existing factors in the current database instance, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
`factor_type_name`	Type of the factor, up to 30 characters in mixed-case, without spaces. To find existing factor types, query the `DBA_DV_FACTOR_TYPE` view, described in "DVSYS.DBA_DV_FACTOR_TYPE View" on page 22-8.
`description`	Description of the purpose of the factor, up to 1024 characters in mixed-case.

Table 16–7 (Cont.) CREATE_FACTOR Parameters

Parameter	Description
`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 `DVSYS.DBA_DV_RULE_SET` view, described in Chapter 22, "Oracle Database Vault Data Dictionary Views.". See also "Assigning a Rule Set to a Factor" on page 8-10 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" on page 8-8 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" on page 8-9 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" on page 8-6 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" on page 8-8 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" on page 8-8 for more information.

Table 16–7 (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" on page 8-11 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" on page 8-10 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 16–8 CREATE_FACTOR_TYPE Parameters

Parameter	Description
`name`	Factor type name, up to 30 characters in mixed-case, without spaces. To find existing factor types, query the `DVSYS.DBA_DV_FACTOR_TYPE` view, described in "DVSYS.DBA_DV_FACTOR_TYPE View" on page 22-8.
`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 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 16–9 CREATE_IDENTITY Parameters

Parameter	Description
`factor_name`	Factor name. To find existing factors, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
`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" on page 8-13 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;
```

```
/
```

CREATE_IDENTITY_MAP Procedure

The `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).

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 16–10 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 `DVSYS.DBA_DV_FACTOR` view, described in Chapter 22, "Oracle Database Vault Data Dictionary Views."
`identity_factor_value`	Value the factor assumes if the identity map evaluates to `TRUE`. To find existing factor identities, query the `DVSYS.DBA_DV_IDENTITY` view, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9. To find current factor identity mappings, use `DVSYS.DBA_DV_IDENTITY_MAP`, described in "DVSYS.DBA_DV_IDENTITY_MAP View" on page 22-9.
`parent_factor_name`	The parent factor link to which the map is related. To find existing parent-child factor mappings, query the `DVSYS.DBA_DV_IDENTITY_MAP` view, described in "DVSYS.DBA_DV_IDENTITY_MAP View" on page 22-9.
`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;
/
```

DELETE_FACTOR Procedure

The `DELETE_FACTOR` procedure deletes a factor.

Syntax

```
DBMS_MACADM.DELETE_FACTOR(
  factor_name IN VARCHAR2);
```

Parameters

Table 16–11 DELETE_FACTOR Parameter

Parameter	Description
`factor_name`	Factor name. To find existing factors in the current database instance, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.

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 16–12 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 `DVSYS.DBA_DV_FACTOR_LINK` view, described in "DVSYS.DBA_DV_FACTOR_LINK View" on page 22-8.
`child_factor_name`	Factor name.

Example

```
BEGIN
 DBMS_MACADM.DELETE_FACTOR_LINK(
  parent_factor_name => 'HQ_ClientID',
  child_factor_name  => 'Div1_ClientID');
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 16–13 DELETE_FACTOR_TYPE Parameters

Parameter	Description
name	Factor type name. To find existing factor types, query the DVSYS.DBA_DV_FACTOR_TYPE view, described in "DVSYS.DBA_DV_FACTOR_TYPE View" on page 22-8.

Example

```
EXEC DBMS_MACADM.DELETE_FACTOR_TYPE('Sector2Instance');
```

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 16–14 DELETE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
value	Identity value associated with the factor. To find the identities for each factor in the current database instance, query the DVSYS.DBA_DV_IDENTITY view, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9.

Example

```
BEGIN
 DBMS_MACADM.DELETE_IDENTITY(
  factor_name => 'Sector2_ClientID',
  value       => 'intranet');
END;
/
```

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 16–15 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 `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
`identity_factor_value`	Value the factor assumes if the identity map evaluates to `TRUE`. To find existing factor identities, query the `DVSYS.DBA_DV_IDENTITY` view, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9. To find current factor identity mappings, query `DVSYS.DBA_DV_IDENTITY_MAP`, described in "DVSYS.DBA_DV_IDENTITY_MAP View" on page 22-9.
`parent_factor_name`	The parent factor link to which the map is related. To find existing parent-child factors, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR_LINK View" on page 22-8.
`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 16–16 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$DATABASE_DOMAIN` function, described in "Oracle Database Vault DVF PL/SQL Factor Functions" on page 16-24.
`domain_host`	Oracle Real Application Clusters host name being that was added to the domain. To find the host name for a specified database, run the `DVF.F$DATABASE_HOSTNAME` function, described in "Oracle Database Vault DVF PL/SQL Factor Functions" on page 16-24.

Example

```
BEGIN
 DBMS_MACADM.DROP_DOMAIN_IDENTITY(
  domain_name  => 'example',
  domain_host  => 'mydom_host');
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 16–17 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;
/
```

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 16–18 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;
/
```

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 16–19 RENAME_FACTOR Parameters

Parameter	Description
`factor_name`	Current factor name. To find existing factors in the current database instance, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
`new_factor_name`	New factor name, up to 28 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 16–20 RENAME_FACTOR_TYPE Parameters

Parameter	Description
old_name	Current factor type name. To find existing factor types in the current database instance, query the DVSYS.DBA_DV_FACTOR_TYPE view, described in "DVSYS.DBA_DV_FACTOR_TYPE View" on page 22-8.
new_name	New factor type name, up to 30 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 16–21 UPDATE_FACTOR

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.

Table 16–21 (Cont.) UPDATE_FACTOR

Parameter	Description
factor_type_name	Factor type name. To find existing factor types, query the DVSYS.DBA_DV_FACTOR_TYPE view, described in "DVSYS.DBA_DV_FACTOR_TYPE View" on page 22-8.
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 DVSYS.DBA_DV_RULE_SET view, described in Chapter 22, "Oracle Database Vault Data Dictionary Views." See also "Assigning a Rule Set to a Factor" on page 8-10 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" on page 8-8 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" on page 8-9 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" on page 8-6 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 DVSYS.DBA_DV_FACTOR data dictionary view. See "Setting the Oracle Label Security Labeling Information for a Factor" on page 8-8 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 ▪ 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 DVSYS.DBA_DV_FACTOR data dictionary view. See "Setting the Evaluation Information for a Factor" on page 8-8 for more information.

Table 16–21 (Cont.) UPDATE_FACTOR

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. The default for `audit_options` is the previously set value, which you can find by querying the `DVSYS.DBA_DV_FACTOR` data dictionary view. See "Setting Audit Options for a Factor" on page 8-11 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 `DVSYS.DBA_DV_FACTOR` data dictionary view. See "Setting Error Options for a Factor" on page 8-10 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 16–22 UPDATE_FACTOR_TYPE Parameters

Parameter	Description
`name`	Factor type name. To find existing factor types in the current database instance, query the `DVSYS.DBA_DV_FACTOR_TYPE` view, described in "DVSYS.DBA_DV_FACTOR_TYPE View" on page 22-8.
`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 16–23 UPDATE_IDENTITY Parameters

Parameter	Description
`factor_name`	Factor name. To find existing factors in the current database instance, query the `DBSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6. To find factors that have identities, query `DVSYS.DBA_DV_IDENTITY`, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9.
`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" on page 8-13 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

You can use a set of standalone procedures and functions to manage factors.

About the Oracle Database Vault Run-Time PL/SQL Procedures and Functions

Oracle Database Vault provides a set of procedural interfaces to administer various Database Vault security options and manage Database Vault security enforcements.

There are also 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.

Table 16–24 lists the default run-time PL/SQL procedures and functions for factors.

Table 16–24 Run-Time PL/SQL Procedures and Functions

Procedure or Function	Parameter
SET_FACTOR Procedure	Sets a factor
GET_FACTOR Function	Retrieves a factor
GET_TRUST_LEVEL Function	Retrieves the trust level assigned to a factor
GET_TRUST_LEVEL_FOR_IDENTITY Function	Retrieves the trust level for a specified factor and an identity
ROLE_IS_ENABLED Function	Checks whether the specified database role is enabled
GET_FACTOR_LABEL Function	Retrieves the label for the specified factor when the factor has a label assigned to it for the specified Oracle Label Security policy

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

```
DVSYS.SET_FACTOR(
  p_factor IN VARCHAR2,
  p_value  IN VARCHAR2);
```

Parameters

Table 16–25 SET_FACTOR Parameters

Parameter	Description
`p_factor`	Factor name. To find existing factors in the current database instance, query the `DVSYS.DBA_DV_FACTOR` data dictionary view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
`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 `DVSYS.DBA_DV_IDENTITY` data dictionary view, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9.

Example

```
EXECUTE DVSYS.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 enables the `F$` functions in the `DVF` schema. This function is available (to execute) to the general database account population.

Syntax

```
DVSYS.GET_FACTOR(
  p_factor IN VARCHAR2)
RETURN VARCHAR2;
```

Parameter

Table 16–26 GET_FACTOR Parameter

Parameter	Description
`p_factor`	Factor name. To find existing factors in the current database instance, query the `DVSYS.DBA_DV_FACTOR` data dictionary view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Factor Identity',
  rule_expr => 'DVSYS.GET_FACTOR(''Sector2_ClientID'')');
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" on page 8-13 for a listing of the available trust levels.

Syntax

```
DVSYS.GET_TRUST_LEVEL(
  p_factor IN VARCHAR2)
RETURN VARCHAR2;
```

Parameter

Table 16–27 GET_TRUST_LEVEL Parameter

Parameter	Description
`p_factor`	Factor name. To find existing factors in the current database instance, query the `DVSYS.DBA_DV_FACTOR` data dictionary view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Trust Level',
  rule_expr => 'DVSYS.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" on page 8-13 for a listing of the available trust levels.

Syntax

```
DVSYS.GET_TRUST_LEVEL_FOR_IDENTITY(
  p_factor   IN VARCHAR2,
  p_identity IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 16–28 GET_TRUST_LEVEL_FOR_IDENTITY Parameters

Parameter	Description
`p_factor`	Factor name. To find existing factors in the current database instance, query the `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6.
`p_identity`	Identity value. To find the identities for each factor in the current database instance, use the `DVSYS.DBA_DV_IDENTITY` data dictionary view, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Identity Trust Level',
  rule_expr => 'DVSYS.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

```
DVSYS.ROLE_IS_ENABLED(
  p_role IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 16–29 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*. ■ `DVSYS.DBA_DV_REALM_AUTH`: Finds the authorization of a particular role. See "DVSYS.DBA_DV_REALM View" on page 22-14. ■ `DVSYS.DBA_DV_ROLE`: Finds existing secure application roles used in privilege management. See "DVSYS.DBA_DV_ROLE View" on page 22-16.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check if SYSADM Role Is Enabled',
  rule_expr => 'DVSYS.ROLE_IS_ENABLED(''SYSADM'')');
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

```
DVSYS.GET_FACTOR_LABEL(
  p_factor      IN VARCHAR2,
  p_policy_name IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 16–30 GET_FACTOR_LABEL Parameters

Parameter	Description
`p_factor`	Factor name. To find the available factors in the current database instance, query the `DVSYS.DBA_DV_FACTOR` data dictionary view. To find factors that are associated with Oracle Label Security policies, use `DVSYS.DBA_DV_MAC_POLICY_FACTOR`. See "DVSYS.DBA_DV_FACTOR View" on page 22-6 and "DVSYS.DBA_DV_MAC_POLICY_FACTOR View" on page 22-11.
`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: ■ `DVSYS.DBA_DV_MAC_POLICY`: Lists Oracle Label Security policies defined in the current database instance. See "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10. ■ `DVSYS.DBA_DV_MAC_POLICY_FACTOR`: Lists the factors that are associated with Oracle Label Security policies for the current database instance. See "DVSYS.DBA_DV_MAC_POLICY_FACTOR View" on page 22-11. ■ `DVSYS.DBA_DV_POLICY_LABEL`: Lists the Oracle Label Security label for each factor identifier in the `DVSYS.DBA_DV_IDENTITY` view for each policy. See "DVSYS.DBA_DV_POLICY_LABEL View" on page 22-12.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get the ClientID Factor Label',
  rule_expr => 'DVSYS.GET_FACTOR_LABEL(''Sector2_ClientID'', ''Access
Locations'')');
END;
/
```

Oracle Database Vault DVF PL/SQL Factor Functions

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.

About the 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. 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;
```

Table 16–31 lists the default factor functions.

Table 16–31 Installed Oracle Database Vault Factor Functions

DVF Factor Function	Description
F$AUTHENTICATION_METHOD Function	Returns the method of authentication in VARCHAR2 data type. In the list that follows, the type of user is followed by the method returned
F$CLIENT_IP Function	Returns the IP address of the computer from which the client is connected
F$DATABASE_DOMAIN Function	Returns the domain of the database as specified in the DB_DOMAIN initialization parameter
F$DATABASE_HOSTNAME Function	Returns the host name of the computer on which the database instance is running
F$DATABASE_INSTANCE Function	Returns the database instance identification number of the current database instance
F$DATABASE_IP Function	Returns the IP address of the computer on which the database instance is running
F$DATABASE_NAME Function	Returns the name of the database as specified in the DB_NAME initialization parameter
F$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
F$ENTERPRISE_IDENTITY Function	Returns the enterprise-wide identity for a user
F$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 USER or ALTER USER syntax.
F$LANG Function	Returns the ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter
F$LANGUAGE Function	Returns the language and territory currently used by your session, in VARCHAR2 data type, along with the database character set
F$MACHINE Function	Returns the computer (host) name for the database client that established the database session.

Table 16–31 (Cont.) Installed Oracle Database Vault Factor Functions

DVF Factor Function	Description
F$NETWORK_PROTOCOL Function	Returns the network protocol being used for communication, as specified in the `PROTOCOL=`*`protocol`* portion of the connect string
F$PROXY_ENTERPRISE_IDENTITY Function	Returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user
F$SESSION_USER Function	Returns the database user name by which the current user is authenticated

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

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client IP Address',
  rule_expr => 'DVF.F$CLIENT_IP BETWEEN ''192.0.2.10'' AND ''192.0.2.20''');
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;
/
```

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$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;
/
```

17

Oracle Database Vault Secure Application Role APIs

You can use the DBMS_MACADM and DBMS_MACSEC_ROLES PL/SQL packages to manage Database Vault secure application roles.

Topics:

- DBMS_MACADM Secure Application Role Procedures
- DBMS_MACSEC_ROLES Secure Application Role Procedure and Function

See Also:

- Chapter 9, "Configuring Secure Application Roles for Oracle Database Vault," for detailed information about realms
- Chapter 19, "Oracle Database Vault Utility APIs," for a set of general-purpose utility procedures that you can use with the secure application role procedures and functions

DBMS_MACADM Secure Application Role Procedures

The DBMS_MACADM package enables you to create, delete, rename, and update Oracle Database Vault secure application roles.

About the DBMS_MACADM Secure Application Role Procedures

Table 17–1 lists procedures within the DBMS_MACADM package that you can use to configure Oracle Database Vault secure application roles. Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures.

Table 17–1 DBMS_MACADM Secure Application Role Configuration 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.
UPDATE_ROLE Procedure	Updates a Oracle Database Vault secure application role

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 17–2 CREATE_ROLE Parameters

Parameter	Description
`role_name`	Role name, up to 30 characters, with no spaces. To find existing secure application roles in the current database instance, query the `DVSYS.DBA_DV_ROLE` view, described in "DVSYS.DBA_DV_ROLE View" on page 22-16.
`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 `DVSYS.DBA_DV_RULE_SET` view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.

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 17–3 DELETE_ROLE Parameter

Parameter	Description
`role_name`	Role name. To find existing secure application roles in the current database instance, query the `DVSYS.DBA_DV_ROLE` view, described in "DVSYS.DBA_DV_ROLE View" on page 22-16.

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 17–4 RENAME_ROLE Parameters

Parameter	Description
`role_name`	Current role name. To find existing secure application roles in the current database instance, query the `DVSYS.DBA_DV_ROLE` view, described in "DVSYS.DBA_DV_ROLE View" on page 22-16.
`new_role_name`	Role name, up to 30 characters, with no spaces. Ensure that this name follows the standard Oracle naming conventions for role creation described in *Oracle Database SQL Language Reference*.

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 17–5 UPDATE_ROLE Parameters

Parameter	Description
`role_name`	Role name. To find existing secure application roles in the current database instance, query the `DVSYS.DBA_DV_ROLE` view, described in "DVSYS.DBA_DV_ROLE View" on page 22-16.
`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 `DVSYS.DBA_DV_ROLE` data dictionary view.

Table 17–5 (Cont.) UPDATE_ROLE Parameters

Parameter	Description
`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 `DVSYS.DBA_DV_RULE_SET` view, described in "DVSYS.DBA_DV_RULE_SET View" on page 22-17.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_ROLE(
  role_name      => 'SECT2_SYSADMIN',
  enabled        => DBMS_MACUTL.G_YES,
  rule_set_name  => 'System Access Controls');
END;
/
```

DBMS_MACSEC_ROLES Secure Application Role Procedure and Function

The `DBMS_MACSEC_ROLES` PL/SQL package enables you check if a user is authorized to use an Oracle Database Vault secure application role and it enables you to set secure application roles.

About the DBMS_MACSEC_ROLES Package

You can modify your applications to use the procedures within the `DBMS_MACSEC_ROLES` package to check the authorization for a user or to set an Oracle Database Vault secure application role. The `DBMS_MACSEC_ROLES` package is available to all users.

Table 17–6 lists the `DBMS_MACSEC_ROLES` package function and procedure.

Table 17–6 DBMS_MACSEC_ROLES Oracle Label Security Configuration Procedures

Function or Procedure	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.

CAN_SET_ROLE Function

The `CAN_SET_ROLE` function checks whether the user invoking the method is authorized to use the specified 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 17–7 CAN_SET_ROLE Parameter

Parameter	Description
`p_role`	Role name. To find existing secure application roles in the current database instance, query the `DVSYS.DBA_DV_ROLE` view, described in "DVSYS.DBA_DV_ROLE View" on page 22-16.

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, including both Oracle Database Vault secure application roles and regular Oracle Database roles. 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 17–8 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 `DVSYS.DBA_DV_ROLE` view, described in "DVSYS.DBA_DV_ROLE View" on page 22-16. 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`).

18 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.

Topics:

About the DBMS_MACADM Oracle Label Security Procedures

Table 18–1 lists procedures within the DBMS_MACADM package that you can use to configure Oracle Label Security policies for Oracle Database Vault. Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures.

Table 18–1 DBMS_MACADM Oracle Label Security Configuration 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

See Also:

- Chapter 10, "Integrating Oracle Database Vault with Other Oracle Products," for detailed information about factors
- "CREATE_RULE Procedure" on page 14-3 for an example of how to create a rule expression that uses the `OLS_LABEL_DOMINATES` function to check the dominance of the current session label
- Chapter 19, "Oracle Database Vault Utility APIs," for a set of general-purpose utility procedures that you can use with Oracle Label Security policy procedures

CREATE_MAC_POLICY Procedure

The `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.

Syntax

```
DBMS_MACADM.CREATE_MAC_POLICY(
  policy_name  IN VARCHAR2,
  algorithm    IN VARCHAR2);
```

Parameters

Table 18–2 CREATE_MAC_POLICY Parameters

Parameter	Description
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DVSYS.DBA_DV_MAC_POLICY` view, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10.
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. Enter the code listed in Table 18–3 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. For more information on label-merging algorithms, see *Oracle Label Security Administrator's Guide*.

Table 18–3 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

Table 18–3 (Cont.) Oracle Label Security Merge Algorithm Codes

Code	Value
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
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 18–4 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 `DVSYS.DBA_DV_FACTOR` view, described in "DVSYS.DBA_DV_FACTOR View" on page 22-6. To find factors that are associated with Oracle Label Security policies, use `DVSYS.DBA_DV_MAC_POLICY_FACTOR`, described in "DVSYS.DBA_DV_MAC_POLICY_FACTOR View" on page 22-11.
`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 `DVSYS.DBA_DV_IDENTITY` view, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9.
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DVSYS.DBA_DV_MAC_POLICY` view, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10.
`label`	Oracle Label Security label name. To find existing policy labels for factor identifiers, query the `DVSYS.DBA_DV_POLICY_LABEL` view, described in "DVSYS.DBA_DV_POLICY_LABEL View" on page 22-12.

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 18–5 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 `DVSYS.DBA_DV_MAC_POLICY` view, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10.

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 18–6 DELETE_POLICY_FACTOR Parameters

Parameter	Description
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DVSYS.DBA_DV_MAC_POLICY` view, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10.
`factor_name`	Name of factor associated with the Oracle Label Security label. To find factors that are associated with Oracle Label Security policies, query `DVSYS.DBA_DV_MAC_POLICY_FACTOR`, described in "DVSYS.DBA_DV_MAC_POLICY_FACTOR View" on page 22-11.

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 18–7 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 `DVSYS.DBA_DV_MAC_POLICY_FACTOR`, described in "DVSYS.DBA_DV_MAC_POLICY_FACTOR View" on page 22-11.
`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 `DVSYS.DBA_DV_IDENTITY` view, described in "DVSYS.DBA_DV_IDENTITY View" on page 22-9.
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DVSYS.DBA_DV_MAC_POLICY` view, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10.
`label`	Oracle Label Security label name. To find existing policy labels for factor identifiers, query the `DVSYS.DBA_DV_POLICY_LABEL` view, described in "DVSYS.DBA_DV_POLICY_LABEL View" on page 22-12.

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 that is used 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 18–8 UPDATE_MAC_POLICY

Parameter	Description
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DVSYS.DBA_DV_MAC_POLICY` view, described in "DVSYS.DBA_DV_MAC_POLICY View" on page 22-10.

Table 18–8 (Cont.) UPDATE_MAC_POLICY

Parameter	Description
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. See Table 18–3 on page 18-2 for listing of the available algorithms. For more information on label-merging algorithms, see *Oracle Label Security Administrator's Guide*.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_MAC_POLICY(
  policy_name  => 'Access Locations',
  algorithm    => 'LUI');
END;
/
```

19 Oracle Database Vault Utility APIs

Oracle Database Vault provides a set of utility APIs in the DBMS_MACUTL PL/SQL package.

Topics:

- About the DBMS_MACUTL Package
- DBMS_MACUTL Constants
- DBMS_MACUTL Package Procedures and Functions

About the DBMS_MACUTL Package

The DBMS_MACUTL package provides a set of general purpose utility procedures and functions that you can use throughout the application code you write for Oracle Database Vault.

This package is available to users who have been granted the DV_OWNER and DV_ADMIN roles.

DBMS_MACUTL Constants

You can use a set of constants, available in the DBMS_MACUTL PL/SQL package.

Topics:

- DBMS_MACUTL Listing of Constants
- Example: Creating a Realm Using DBMS_MACUTL Constants
- Example: Creating a Rule Set Using DBMS_MACUTL Constants
- Example: Creating a Factor Using DBMS_MACUTL Constants

DBMS_MACUTL Listing of Constants

Table 19–1 summarizes constant (that is, fields) descriptions for the DBMS_MACUTL package.

You can use these constants with any of the Oracle Database Vault PL/SQL packages. 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 19–1 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_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.
`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.

Table 19–1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
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.
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.

Table 19–1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
`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.
`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_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

Example 19–1 shows how to use the `G_YES` and `G_REALM_AUDIT_FAIL` `DBMS_MACUTL` constants when creating a realm.

Example 19–1 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

Example 19–2 shows how to use several DBMS_MACUTL constants when creating a rule set.

Example 19–2 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

Example 19–3 shows how to use constants when creating a factor.

Example 19–3 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

You can use the DBMS_MACUTL PL/SQL package to perform tasks such as finding a time value or whether a user has the DVOWNER role or the appropriate privileges.

About the DBMS_MACUTL Package Procedures and Functions

Table 19–2 lists the procedures and functions in the DBMS_MACUTL PL/SQL package. You can use these procedures or functions as standalone code, or within rule expressions. The examples in this section show a mixture of using both.

Table 19–2 DBMS_MACUTL Utility Functions

Procedure or Function	Description
CHECK_DVSYS_DML_ALLOWED Procedure	Checks whether the given user can issue DML commands to access the `DVSYS` objects
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
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_OBJECT_PRIVILEGE Function	Checks whether a user or role may access an object through an object privilege grant.
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)

CHECK_DVSYS_DML_ALLOWED Procedure

The `CHECK_DVSYS_DML_ALLOWED` procedure checks whether the given 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 19–3 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*. ■ DVSYS.DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15. ■ DVSYS.DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "DVSYS.DBA_DV_ROLE View" on page 22-16.

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 19–4 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 DVSYS.DBA_DV_CODE view, described in "DVSYS.DBA_DV_CODE View" on page 22-2.

Table 19–4 (Cont.) GET_CODE_VALUE Parameters

Parameter	Description
`p_code`	ID of the code. This ID is listed when you run the `DVSYS.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 19–5 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 19–6 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 19–7 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 19–8 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 19–9 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 19–10 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 checks whether the character is alphabetic, and then returns a BOOLEAN value. IS_ALPHA returns TRUE if the character is alphabetic.

Syntax

```
DBMS_MACUTL.IS_ALPHA(
  c IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 19–11 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 whether the character is numeric, and then returns a BOOLEAN value. IS_DIGIT returns TRUE if the character is a digit.

Syntax

```
DBMS_MACUTL.IS_DIGIT(
  c IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 19–12 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 determines whether a user is authorized to manage the Oracle Database Vault configuration, and then returns a BOOLEAN value. 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 19–13 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*. ▪ DVSYS.DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15. ▪ DVSYS.DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "DVSYS.DBA_DV_ROLE View" on page 22-16.

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 an indicator regarding whether Oracle Label Security is installed, and then returns a TRUE or FALSE BOOLEAN value. 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 an indicator regarding whether Oracle Label Security is installed, and then returns a Y or N VARCHAR2 value. 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" on page 19-13 for an example.

USER_HAS_OBJECT_PRIVILEGE Function

The USER_HAS_OBJECT_PRIVILEGE function checks whether a user or role can access an object through a single specified object privilege grant, and then returns a BOOLEAN value. 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 19–14   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 "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15. ■ DVSYS.DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "DVSYS.DBA_DV_ROLE View" on page 22-16.

Table 19–14 (Cont.) USER_HAS_OBJECT_PRIVILEGE Parameters

Parameter	Description
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 they ALL_OBJECTS view, described in *Oracle Database Reference*. To find objects that are secured by existing realms, query they DVSYS.DBA_DV_REALM_OBJECT view.
p_privilege	Object privilege, such as, UPDATE. To find privileges for a database account excluding PUBLIC privileges, query they DVSYS.DBA_DV_USER_PRIVS view. To find all privileges for a database account, query the DVSYS.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 checks whether a user has a role privilege, directly or indirectly (through another role), and then returns a BOOLEAN value. 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 19–15 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*. ■ `DVSYS.DBA_DV_REALM_AUTH`: Finds the authorization of a particular user or role. See "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15. ■ `DVSYS.DBA_DV_ROLE`: Finds existing secure application roles used in privilege management. See "DVSYS.DBA_DV_ROLE View" on page 22-16.
`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*. ■ `DVSYS.DBA_DV_REALM_AUTH`: Finds the authorization of a particular user or role. See Chapter 22, "Oracle Database Vault Data Dictionary Views.".

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 checks whether a user has a role privilege, directly or indirectly (through another role), and then returns a `VARCHAR2` value. 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 19–16 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*. ■ `DVSYS.DBA_DV_REALM_AUTH`: Finds the authorization of a particular user or role. See "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15. ■ `DVSYS.DBA_DV_ROLE`: Finds existing secure application roles used in privilege management. See "DVSYS.DBA_DV_ROLE View" on page 22-16.
`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*. ■ `DVSYS.DBA_DV_REALM_AUTH`: Finds the authorization of a particular user or role. See "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15.

USER_HAS_SYSTEM_PRIVILEGE Function

The `USER_HAS_SYSTEM_PRIVILEGE` function checks whether a user has a system privilege, directly or indirectly (through a role), and then returns a `BOOLEAN` value. 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 19–17 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 `DVSYS.DBA_DV_USER_PRIVS` view, described in "DVSYS.DBA_DV_USER_PRIVS View" on page 22-20. To find all privileges for a database account, use `DVSYS.DBA_DV_USER_PRIVS_ALL`, described in "DVSYS.DBA_DV_USER_PRIVS_ALL View" on page 22-21.

Table 19–17 (Cont.) USER_HAS_SYSTEM_PRIVILEGE Parameters

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*. ■ DVSYS.DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See "DVSYS.DBA_DV_REALM_AUTH View" on page 22-15.

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;
/
```

20 Oracle Database Vault General Administrative APIs

The DBMS_MACADM PL/SQL package and the DVSYS.CONFIGURE_DV standalone procedure enable you to you perform general maintenance tasks.

- DBMS_MACADM General System Maintenance Procedures
- DVSYS.CONFIGURE_DV General System Maintenance Procedure

DBMS_MACADM General System Maintenance Procedures

The general system maintenance procedures of the DBMS_MACADM PL/SQL package enable you to you perform tasks such as authorizing users or adding new language to Oracle Database Vault.

About the DBMS_MACADM General System Maintenance Procedures

Table 20–1 lists the procedures within the DBMS_MACADM PL/SQL package that you can use to perform general maintenance activities that require the DV_OWNER role.

Table 20–1 DBMS_MACADM General System Maintenance 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	Grants a user authorization to execute data definition language (DDL) statements on the specified schema
AUTHORIZE_PROXY_USER	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

Table 20–1 (Cont.) DBMS_MACADM General System Maintenance Procedures

Procedure	Description
UNAUTHORIZE_DDL Procedure	Revokes authorization from a user who was granted authorization to execute DDL statements through the `DBMS_MACDM.AUTHORIZE_DDL` procedure
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_AUDIT Procedure	Disables realm, command rule, and rule set auditing of the actions by users who have been granted the `DV_PATCH_ADMIN` role
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_AUDIT Procedure	Enables realm, command rule, and rule set auditing of the actions by users who have been granted the `DV_PATCH_ADMIN` role
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 20–2 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 for Oracle Data Pump Regular Operations in Database Vault" on page 11-6 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 20–3 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%'`

Table 20–3 (Cont.) AUTHORIZE_DATAPUMP_USER

Parameter	Description
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_DDL

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 DVSYS.DBA_DV_DDL_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_DDL(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2);
```

Parameters

Table 20–4 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_PROXY_USER

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).

To find information about users who have been granted this authorization, query the `DVSYS.DBA_DV_PROXY_AUTH` view.

Syntax

```
DBMS_MACADM.AUTHORIZE_PROXY_USER(
  proxy_user   IN VARCHAR2,
  user_name    IN VARCHAR2);
```

Parameters

Table 20–5 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:

```
DBMS_MACADM.AUTHORIZE_PROXY_USER('preston', '%');
```

This example enables proxy user `preston` to proxy database user `dkent` only.

```
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" on page 11-12 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 20–6 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 Oracle Database Vault" on page 11-12.
`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.

See "Authorizing Users for Oracle Data Pump Regular Operations in Database Vault" on page 11-6 for 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 20–7 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 20–8 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 `DVSYS.DVSYS.DBA_DV_DATAPUMP_AUTH` data dictionary view as follows: `SELECT * FROM DVSYS.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_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 `DVSYS.DBA_DV_DDL_AUTH` data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DDL(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2);
```

Parameters

Table 20–9 UNAUTHORIZE_DDL

Parameter	Description
`user_name`	Name of the user from whom you want to revoke DDL authorization.

Table 20–9 (Cont.) UNAUTHORIZE_DDL

Parameter	Description
`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_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 20–10 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.

```
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 20–11 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 `DVSYS.DBA_DV_JOB_AUTH` data dictionary view as follows: `SELECT * FROM DVSYS.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 for a tablespace when Oracle Database Vault is enabled.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_TTS_USER
  uname      IN VARCHAR2,
  tsname     IN VARCHAR2);
```

Parameters

Table 20–12 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.

Table 20–12 (Cont.) UNAUTHORIZE_TTS_USER

Parameter	Description
`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;
```

See Also: Appendix B, "Disabling and Enabling Oracle Database Vault," for detailed information about disabling and enabling Database Vault, including to how to find out if Database Vault is enabled

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 `DVSYS.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;
```

See Also: "Archiving and Purging the Oracle Database Vault Audit Trail" on page A-5

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 been granted 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 `DVSYS.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;
```

See Also:

- "DV_PATCH_ADMIN Database Vault Database Patch Role" on page 12-15
- "ENABLE_DV_PATCH_ADMIN_AUDIT Procedure" on page 20-12

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;
```

See Also: "Using the ORADEBUG Utility with Oracle Database Vault" on page 11-20

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;
```

Parameters

None.

Example

```
EXEC DBMS_MACADM.ENABLE_DV;
```

> **See Also:** Appendix B, "Disabling and Enabling Oracle Database Vault," for detailed information about disabling and enabling Database Vault, including to how to find out if Database Vault is enabled

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 been granted the `DV_PATCH_ADMIN` role, in accordance with the existing audit configuration.

This procedure is designed to audit these users' actions during a patch upgrade. To find if this auditing is enabled or not, query the `DVSYS.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;
```

> **See Also:**
>
> - "DV_PATCH_ADMIN Database Vault Database Patch Role" on page 12-15
> - "DISABLE_DV_PATCH_ADMIN_AUDIT Procedure" on page 20-11

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, these 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 `DVSYS.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;
```

> **See Also:** "Archiving and Purging the Oracle Database Vault Audit Trail" on page A-5

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;
```

> **See Also:** "Using the ORADEBUG Utility with Oracle Database Vault" on page 11-20

DVSYS.CONFIGURE_DV General System Maintenance Procedure

The `DVSYS.CONFIGURE_DV` procedure configures the initial two Oracle Database user accounts, which are granted the `DV_OWNER` and `DV_ACCTMGR` roles, respectively.

Before you run this 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.

You only can run the `DVSYS.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 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 "Oracle Database Vault Accounts" on page 12-19 for guidance.

When you run the `DVSYS.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, deinstall and then reinstall Oracle Database Vault. See the following sections for more information:

- "Deinstalling Oracle Database Vault" on page C-2
- "Reinstalling Oracle Database Vault" on page C-3

Together, the `DVSYS.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 this procedure as user `SYS`. See "Registering Oracle Database Vault with an Oracle Database" on page 3-1 for the process you would use.

Syntax

```
DVSYS.CONFIGURE_DV
  dvowner_uname         IN VARCHAR2,
  dvacctmgr_uname       IN VARCHAR2;
```

Parameters

Table 20–13 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
 DVSYS.CONFIGURE_DV (
   dvowner_uname         => 'dbv_owner',
   dvacctmgr_uname       => 'dbv_acctmgr';
 END;
/
```

21

Oracle Database Vault API Reference

Oracle Database Vault provides a rich set of APIs, both in PL/SQL packages and in standalone procedures.

Topics:

- DBMS_MACADM PL/SQL Package Contents
- DBMS_MACSEC_ROLES PL/SQL Package Contents
- DBMS_MACUTL PL/SQL Package Contents
- DVSYS.CONFIGURE_DV PL/SQL Procedure
- DVF PL/SQL Interface Contents

DBMS_MACADM PL/SQL Package Contents

The procedures and functions within the DBMS_MACADM package allow you to write applications that configure the realms, factors, rule sets, command rules, secure application roles, and Oracle Label Security policies normally configured in Oracle Database Vault Administrator.

The DBMS_MACADM package is available only for users who have been granted the DV_ADMIN or DV_OWNER role.

Table 20–1 lists the contents of the DBMS_MACADM package.

Table 21–1 DBMS_MACADM PL/SQL Package Contents

Procedure or Function	Description
Realm APIs	
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
DELETE_REALM procedure	Deletes a realm, including its related Database Vault configuration information that specifies who is authorized and what objects are protected

Table 21–1 (Cont.) DBMS_MACADM PL/SQL Package Contents

Procedure or Function	Description
`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
Rule Set APIs	
`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
Rule APIs	
`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
Command Rule APIs	
`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
`DELETE_COMMAND_RULE` procedure	Drops a command rule declaration
`UPDATE_COMMAND_RULE` procedure	Updates a command rule declaration
Factor APIs	
`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

Table 21–1 (Cont.) DBMS_MACADM PL/SQL Package Contents

Procedure or Function	Description
`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)
`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
Database Vault Secure Application Role APIs	
`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.
`UPDATE_ROLE` procedure	Updates a Oracle Database Vault secure application role
Oracle Label Security APIs	
`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

Table 21–1 (Cont.) DBMS_MACADM PL/SQL Package Contents

Procedure or Function	Description
`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
General Administrative APIs	
`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_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_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

See Also:

- Chapter 13, "Oracle Database Vault Realm APIs"
- "DBMS_MACADM Rule Set Procedures" on page 14-1
- Chapter 15, "Oracle Database Vault Command Rule APIs"
- "DBMS_MACADM Factor Procedures and Functions" on page 16-1
- "DBMS_MACADM Secure Application Role Procedures" on page 17-1
- Chapter 18, "Oracle Database Vault Oracle Label Security APIs"
- "DBMS_MACADM General System Maintenance Procedures" on page 20-1

DBMS_MACSEC_ROLES PL/SQL Package Contents

The `DBMS_MACSEC_ROLES` package provides one function and one procedure which enable to you to check and set Oracle Database Vault secure application roles. This package is available to the general database account population.

Table 20–2 lists the contents of the `DBMS_MACSEC_ROLES` package.

Table 21–2 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 several constants and utility methods that are commonly used by other Oracle Database Vault packages, such as code/message lookup, error handling, data conversion, and privilege checks.

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 20–3 lists the `DBMS_MACUTL` package contents.

Table 21–3 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

Table 21–3 (Cont.) DBMS_MACUTL PL/SQL Package Contents

Procedure or Function	Description
`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
`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)

See Also: Chapter 19, "Oracle Database Vault Utility APIs" for details about this package

DVSYS.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.

See Also:

- "DVSYS.CONFIGURE_DV General System Maintenance Procedure" on page 20-13
- "Registering Oracle Database Vault with an Oracle Database" on page 3-1 for how to use this procedure to register users

See for more information.

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 21–4 lists the `DVF` factor functions.

Table 21–4 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$DATABASE_IP`	Returns the IP address of the computer on which the database instance is running
`F$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

See Also: "Oracle Database Vault DVF PL/SQL Factor Functions" on page 16-24 for detailed information about these functions

22

Oracle Database Vault Data Dictionary Views

You can find information about the Oracle Database Vault configuration settings by querying a set of Database Vault-specific data dictionary views.

Topics:

- DVSYS.DBA_DV_RULE_SET View
- DVSYS.DBA_DV_RULE_SET_RULE View
- DVSYS.DBA_DV_TTS_AUTH View
- DVSYS.DBA_DV_USER_PRIVS View
- DVSYS.DBA_DV_USER_PRIVS_ALL View
- DVSYS.DV$CONFIGURATION_AUDIT View
- DVSYS.DV$ENFORCEMENT_AUDIT View
- SYS.DV$CONFIGURATION_AUDIT View
- SYS.DV$ENFORCEMENT_AUDIT View

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: Chapter 24, "Oracle Database Vault Reports" if you are interested in running reports on Oracle Database Vault

DVSYS.DBA_DV_CODE View

The `DVSYS.DBA_DV_CODE` data dictionary view lists generic lookup codes for the user interface, error messages, constraint checking, and so on. These codes are used for the user interface, views, and for validating input in a translatable fashion.

For example:

```
SELECT CODE, VALUE FROM DVSYS.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(30)`	`NOT NULL`	Displays one of the code groups that are listed in Table 22–1 on page 22-3
`CODE`	`VARCHAR(30)`	`NOT NULL`	Boolean code used; either `Y` (Yes) or `N` (No).
`VALUE`	`VARCHAR(4000)`		Boolean value used; either `True` if the Boolean code is `Y` or `False` if the Boolean code is `N`.

Column	Datatype	Null	Description
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)		Brief description of the code group.

Table 22–1 describes the possible values from the CODE_GROUP column in the DVSYS.DBA_DV_CODE data dictionary view.

Table 22–1 DVSYS.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 18–3, " Oracle Label Security Merge Algorithm Codes" on page 18-2 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

Table 22–1 (Cont.) DVSYS.DBA_DV_CODE View CODE_GROUP Values

CODE_GROUP Name	Description
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

DVSYS.DBA_DV_COMMAND_RULE View

The DVSYS.DBA_DV_COMMAND_RULE data dictionary view lists the SQL statements that are protected by command rules.

See Chapter 7, "Configuring Command Rules," for more information about command rules.

For example:

```
SELECT COMMAND, RULE_SET_NAME FROM DVSYS.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(30)	NOT NULL	Name of the command rule. For a list of default command rules, see "Default Command Rules" on page 7-3.
RULE_SET_NAME	VARCHAR(90)	NOT NULL	Name of the rule set associated with this command rule.
OBJECT_OWNER	VARCHAR(30)	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.

DVSYS.DBA_DV_DATAPUMP_AUTH View

The DVSYS.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" on page 11-4 for more information.

For example:

```
SELECT * FROM DVSYS.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
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)

DVSYS.DBA_DV_DDL_AUTH View

The DVSYS.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 DVSYS.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" on page 20-4
- "UNAUTHORIZE_DDL Procedure" on page 20-7

DVSYS.DBA_DV_DICTIONARY_ACCTS View

The DVSYS.DBA_DV_DICTIONARY_ACCTS data dictionary view indicates whether users can directly log into the DVSYS and DVF schema accounts.

For example:

```
SELECT * FROM DVSYS.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

DVSYS.DBA_DV_FACTOR View

The DVSYS.DBA_DV_FACTOR data dictionary view lists the existing factors in the current database instance.

For example:

```
SELECT NAME, GET_EXPR FROM DVSYS.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

- DVSYS.DBA_DV_FACTOR_LINK View
- DVSYS.DBA_DV_FACTOR_TYPE View

Column	Datatype	Null	Description
NAME	VARCHAR2(30)	NOT NULL	Name of the factor. See "Default Factors" on page 8-2 for a list of default factors.
DESCRIPTION	VARCHAR2(4000)		Description of the factor.
FACTOR_TYPE_NAME	VARCHAR2(90)	NOT NULL	Category of the factor, which is used to classify the purpose of the factor.
ASSIGN_RULE_SET_NAME	VARCHAR2(90)		Rule set used to control the identify of the factor.
GET_EXPR	VARCHAR2(1024)		PL/SQL expression that retrieves the identity of a factor.
VALIDATE_EXPR	VARCHAR2(1024)		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
IDENTIFIED_BY_MEANING	VARCHAR2(4000)		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

Column	Datatype	Null	Description
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)		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)		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.
FAIL_OPTIONS_MEANING	VARCHAR2(4000)		Provides a text description for the corresponding value in the FAIL_OPTIONS column. Possible values are: ■ Show Error Message ■ Do Not Show Error Message:

DVSYS.DBA_DV_FACTOR_LINK View

The DVSYS.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 DVSYS.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

- DVSYS.DBA_DV_FACTOR View
- DVSYS.DBA_DV_FACTOR_TYPE View

Column	Datatype	Null	Description
PARENT_FACTOR_NAME	VARCHAR(30)	NOT NULL	Name of the parent factor.
CHILD_FACTOR_NAME	VARCHAR(30)	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)

DVSYS.DBA_DV_FACTOR_TYPE View

The DVSYS.DBA_DV_FACTOR_TYPE data dictionary view lists the names and descriptions of factor types used in the system.

For example:

```
SELECT * FROM DVSYS.DBA_DV_FACTOR_TYPE WHERE NAME = 'Hostname';
```

Output similar to the following appears:

```
NAME      DESCRIPTION
--------- ----------------------------------------------------------------------
Time      Time-based factor
```

Related Views

- DVSYS.DBA_DV_FACTOR View
- DVSYS.DBA_DV_FACTOR_LINK View

Column	Datatype	Null	Description
NAME	VARCHAR(90)	NOT NULL	Name of the factor type.

Column	Datatype	Null	Description
DESCRIPTION	VARCHAR(1024)		Description of the factor type.

DVSYS.DBA_DV_IDENTITY View

The DVSYS.DBA_DV_IDENTITY data dictionary view lists the identities for each factor.

For example:

```
SELECT * FROM DVSYS.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

- DVSYS.DBA_DV_FACTOR View
- DVSYS.DBA_DV_IDENTITY_MAP View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(30)	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.

DVSYS.DBA_DV_IDENTITY_MAP View

The DVSYS.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 DVSYS.DBA_DV_IDENTITY_MAP;
```

Output similar to the following appears:

```
FACTOR_NAME      IDENTITY_VALUE
---------------- --------------------
Sector2_Program  Accounting-Sensitive
```

Related Views

- DVSYS.DBA_DV_FACTOR View
- DVSYS.DBA_DV_IDENTITY View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(30)	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(30)	NOT NULL	Descriptive name of the operation in the OPERATION_VALUE column.
OPERATION_VALUE	VARCHAR(4000)		Relational operator for the identity map (for example, <, >, =, and so on).
OPERAND1	VARCHAR(1024)		Left operand for the relational operator; refers to the low value you enter.
OPERAND2	VARCHAR(1024)		Right operand for the relational operator; refers to the high value you enter.
PARENT_FACTOR_NAME	VARCHAR(30)		The parent factor link to which the map is related.
CHILD_FACTOR_NAME	VARCHAR(30)		The child factor link to which the map is related.
LABEL_IND	VARCHAR(1)		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)

DVSYS.DBA_DV_JOB_AUTH View

The DVSYS.DBA_DV_JOB_AUTH data dictionary view lists the authorizations for using Oracle Scheduler in an Oracle Database Vault environment.

For example:

```
SELECT * FROM DVSYS.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

DVSYS.DBA_DV_MAC_POLICY View

The DVSYS.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 DVSYS.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

- DVSYS.DBA_DV_MAC_POLICY_FACTOR View
- DVSYS.DBA_DV_POLICY_LABEL View

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(30)	NOT NULL	Name of the policy.
ALGORITHM_CODE	VARCHAR(30)	NOT NULL	Merge algorithm code used for the policy. See Table 18–3 on page 18-2 for a listing of algorithm codes.
ALGORITHM_MEANING	VARCHAR(4000)		Provides a text description for the corresponding value in the ALGORITHM_CODE column. See Table 18–3 on page 18-2 for a listing of algorithm code descriptions.
ERROR_LABEL	VARCHAR(4000)		Label specified for initialization errors, to be set when a configuration error or run-time error occurs during session initialization.

DVSYS.DBA_DV_MAC_POLICY_FACTOR View

The DVSYS.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 DVSYS.DBA_DV_MAC_POLICY_FACTOR;
```

Output similar to the following appears:

```
FACTOR_NAME    MAC_POLICY_NAME
-------------- ------------------
App_Host_Name  Access Locations
```

Related Views

- DVSYS.DBA_DV_MAC_POLICY View
- DVSYS.DBA_DV_POLICY_LABEL View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(30)	NOT NULL	Name of the factor.
MAC_POLICY_NAME	VARCHAR(30)	NOT NULL	Name of the Oracle Label Security policy associated with this factor.

DVSYS.DBA_DV_ORADEBUG View

The DVSYS.DBA_DV_ORADEBUG data dictionary view indicates whether users can use the ORADEBUG utility in an Oracle Database Vault environment.

For example:

```
SELECT * FROM DVSYS.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

DVSYS.DBA_DV_PATCH_ADMIN_AUDIT View

The DVSYS.DBA_DV_PATCH_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 DVSYS.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" on page 20-12
- "DISABLE_DV_PATCH_ADMIN_AUDIT Procedure" on page 20-11

DVSYS.DBA_DV_POLICY_LABEL View

The DVSYS.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 DVSYS.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

- DVSYS.DBA_DV_MAC_POLICY View
- DVSYS.DBA_DV_MAC_POLICY_FACTOR View

Column	Datatype	Null	Description
IDENTITY_VALUE	VARCHAR(1024)	NOT NULL	Name of the factor identifier.
FACTOR_NAME	VARCHAR(30)	NOT NULL	Name of the factor associated with the factor identifier.
POLICY_NAME	VARCHAR(30)	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.

DVSYS.DBA_DV_PROXY_AUTH View

The DVSYS.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 DVSYS.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" on page 20-4
- "UNAUTHORIZE_PROXY_USER Procedure" on page 20-8

DVSYS.DBA_DV_PUB_PRIVS View

The DVSYS.DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in the Oracle Database Vault Administrator (DV_ADMIN). See also "Privilege Management - Summary Reports" on page 24-10.

For example:

```
SELECT USERNAME, ACCESS_TYPE FROM DVSYS.DBA_DV_PUB_PRIVS WHERE USERNAME = 'OE';
```

Output similar to the following appears:

```
USERNAME    ACCESS_TYPE
----------- -----------------
```

```
OE          PUBLIC
```

Related Views

- DVSYS.DBA_DV_USER_PRIVS View
- DVSYS.DBA_DV_USER_PRIVS_ALL View
- DVSYS.DBA_DV_ROLE View

Column	Datatype	Null	Description
USERNAME	VARCHAR(30)	NOT NULL	Database schema in the current database instance.acces
ACCESS_TYPE	VARCHAR(30)		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(30)	NOT NULL	Owner of the database schema to which the USERNAME user has been granted privileges.
OBJECT_NAME	VARCHAR(30)	NOT NULL	Name of the object within the schema listed in the OWNER column.

DVSYS.DBA_DV_REALM View

The DVSYS.DBA_DV_REALM data dictionary view lists the realms created in the current database instance.

For example:

```
SELECT NAME, AUDIT_OPTIONS, ENABLED FROM DVSYS.DBA_DV_REALM
  WHERE AUDIT_OPTIONS = '1';
```

Output similar to the following appears:

```
NAME                          AUDIT_OPTIONS    ENABLED
----------------------------- ---------------- --------
Performance Statistics Realm  1                Y
```

Related Views

- DVSYS.DBA_DV_REALM_AUTH View
- DVSYS.DBA_DV_REALM_OBJECT View

Column	Datatype	Null	Description
NAME	VARCHAR(90)	NOT NULL	Names of the realms created. See"Default Realms" on page 5-4 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.

Column	Datatype	Null	Description
ENABLED	VARCHAR(1)	NOT NULL	Specifies whether realm checking is enabled. Y (Yes) indicates it is enabled; N (No) indicates it is not.

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, whether the realm is a mandatory realm, and so on. It 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

- DVSYS.DBA_DV_REALM View

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	ID number of the realm
NAME	VARCHAR2(90)	NOT NULL	Name of the realm
DESCRIPTION	VARCHAR2(1024)		Description of the realm
AUDIT_OPTIONS	NUMBER	NOT NULL	Audit options set for the realm. See audit_options in Table 13–10 on page 13-9 for a description of the possible values.
ENABLED	VARCHAR2(1)	NOT NULL	Whether the realm has been enabled. See enabled in Table 13–10 on page 13-9 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 13–10 on page 13-9 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.
UPDATE_BY	VARCHAR2(128)	NULL	User who last updated the realm
UPDATE_DATE	DATE		Date on which the realm was last updated, in the following format: <need format>

DVSYS.DBA_DV_REALM_AUTH View

The DVSYS.DBA_DV_REALM_AUTH data dictionary view lists the authorization of a named database user account or database role (GRANTEE) to access realm objects in a particular realm. See "About Realm Authorization" on page 5-9 for more information.

For example:

```
SELECT REALM_NAME, GRANTEE, AUTH_RULE_SET_NAME FROM DVSYS.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

- DVSYS.DBA_DV_REALM View
- DVSYS.DBA_DV_REALM_OBJECT View

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(90)	NOT NULL	Name of the realm.
GRANTEE	VARCHAR(30)	NOT NULL	User or role name to authorize as owner or participant.
AUTH_RULE_SET_NAME	VARCHAR(90)		Rule set to check before authorizing. If the rule set evaluates to TRUE, then the authorization is allowed.
AUTH_OPTIONS	VARCHAR(4000)		Type of realm authorization: either Participant or Owner.

DVSYS.DBA_DV_REALM_OBJECT View

The DVSYS.DBA_DV_REALM_OBJECT data dictionary view lists the database schemas, or subsets of schemas with specific database objects contained therein, that are secured by the realms. See "About Realm-Secured Objects" on page 5-9 for more information.

For example:

```
SELECT REALM_NAME, OWNER, OBJECT_NAME FROM DVSYS.DBA_DV_REALM_OBJECT;
```

Output similar to the following appears:

```
REALM_NAME                   OWNER    OBJECT_NAME
---------------------------- -------- -----------
Performance Statistics Realm OE       ORDERS
```

Related Views

- DVSYS.DBA_DV_REALM View
- DVSYS.DBA_DV_REALM_AUTH View

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(90)	NOT NULL	Name of the realm.
OWNER	VARCHAR(90)	NOT NULL	Database schema owner who owns the realm.
OBJECT_NAME	VARCHAR(90)	NOT NULL	Name of the object the realm protects.
OBJECT_TYPE	VARCHAR(90)	NOT NULL	Type of object the realm protects, such as a database table, view, index, or role.

DVSYS.DBA_DV_ROLE View

The DVSYS.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 DVSYS.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

- DVSYS.DBA_DV_PUB_PRIVS View
- DVSYS.DBA_DV_USER_PRIVS View
- DVSYS.DBA_DV_USER_PRIVS_ALL View

Column	Datatype	Null	Description
ROLE	VARCHAR(30)	NOT NULL	Name of the secure application role.
RULE_NAME	VARCHAR(90)	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. Y (Yes) enables the role; N (No) disables it.

DVSYS.DBA_DV_RULE View

The DVSYS.DBA_DV_RULE data dictionary view lists the rules that have been defined.

For example:

```
SELECT * FROM DVSYS.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

- DVSYS.DBA_DV_RULE_SET View
- DVSYS.DBA_DV_RULE_SET_RULE View

Column	Datatype	Null	Description
NAME	VARCHAR(90)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression for the rule.

DVSYS.DBA_DV_RULE_SET View

The DVSYS.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 DVSYS.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

- DVSYS.DBA_DV_RULE View
- DVSYS.DBA_DV_RULE_SET_RULE View

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(90)	NOT NULL	Name of the rule set.
DESCRIPTION	VARCHAR(1024)		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)		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)		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)		Error message for failure that is associated with the fail code listed in the FAIL_CODE column.
FAIL_CODE	VARCHAR(10)		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.

Column	Datatype	Null	Description
HANDLER	VARCHAR(1024)		Name of the PL/SQL function or procedure that defines the custom event handler logic.
IS_STATIC	VARCHAR2(5)		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.

DVSYS.DBA_DV_RULE_SET_RULE View

The DVSYS.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 DVSYS.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

- DVSYS.DBA_DV_RULE View
- DVSYS.DBA_DV_RULE_SET View

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(90)	NOT NULL	Name of the rule set that contains the rule.
RULE_NAME	VARCHAR(90)	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.

DVSYS.DBA_DV_TTS_AUTH View

The DVSYS.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 in an Oracle Database Vault environment. See "Using Oracle Data Pump with Oracle Database Vault" on page 11-4 for more information.

For example:

```
SELECT * FROM DVSYS.DBA_DV_TTS_AUTH;
```

Output similar to the following appears:

```
GRANTEE  TSNAME
-------- --------
DB_MGR   HR_TS
```

Related Views

- DVSYS.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

DVSYS.DBA_DV_USER_PRIVS View

The DVSYS.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 DVSYS.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

- DVSYS.DBA_DV_PUB_PRIVS View
- DVSYS.DBA_DV_ROLE View
- DVSYS.DBA_DV_USER_PRIVS_ALL View

Column	Datatype	Null	Description
USERNAME	VARCHAR(30)	NOT NULL	Name of the database schema account in which privileges have been defined.
ACCESS_TYPE	VARCHAR(30)		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(30)	NOT NULL	Name of the database user account.
OBJECT_NAME	VARCHAR(30)	NOT NULL	Name of the PL/SQL function or procedure used to define privileges.

DVSYS.DBA_DV_USER_PRIVS_ALL View

The `DVSYS.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 DVSYS.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

- DVSYS.DBA_DV_PUB_PRIVS View
- DVSYS.DBA_DV_ROLE View
- DVSYS.DBA_DV_USER_PRIVS View

Column	Datatype	Null	Description
`USERNAME`	`VARCHAR(30)`		Name of the database schema account in which privileges have been defined.
`ACCESS_TYPE`	`VARCHAR(30)`		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)`		Privilege granted to the user listed in the `USERNAME` column.
`OWNER`	`VARCHAR(30)`		Name of the database user account.
`OBJECT_NAME`	`VARCHAR(30)`		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 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

- SYS.DV$CONFIGURATION_AUDIT View

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR(255)		Operating system login user name of the user whose actions were audited
USERNAME	VARCHAR(128)		Name of the database user whose actions were audited
USERHOST	VARCHAR2(128)		Client computer name
TERMINAL	VARCHAR2(30)		Identifier for the user's terminal
TIMESTAMP	DATA		Date and time of creation of the audit trail entry (in the local database session time zone)
OWNER	VARCHAR2(128)		Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR2(128)		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 22–2 on page 22-23 for a listing of the possible actions.
ACTION_NAME	VARCHAR2(30)		Name of the action type corresponding to the numeric code in the ACTION column. See Table 22–2 on page 22-23 for a listing of the possible actions.
ACTION_OBJECT_ID	NUMBER		The unique identifier of the record in the table specified under OBJ_NAME
ACTION_OBJECT_NAME	VARCHAR2(128)		The unique name or natural key of the record in the table specified under OBJ_NAME
ACTION_COMMAND	VARCHAR2(4000)		The SQL text of the command procedure that was executed that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)		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		The unique identifier of the rule set that was executing and caused the audit event to trigger
RULE_SET_NAME	VARCHAR2(90)		The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER		Not used
RULE_NAME	VARCHAR2(128)		Not used
FACTOR_CONTEXT	VARCHAR2(4000)		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)		Text comment on the audit trail entry, providing more information about the statement audited

Column	Datatype	Null	Description
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		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		Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)		Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER		Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)		Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(128)		Database login user name of the user whose actions were audited
CREATE_DATE	DATE		Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(128)		Same as CREATED_BY column value
UPDATE_DATE	DATE		Same as UPDATED_BY column value
GRANTEE	VARCHAR2(128)		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)		Indicates whether the configuration was enabled

Table 22–2 describes the possible values for the ACTION column of the DVSYS.DV$CONFIGURATION_AUDIT view.

Table 22–2 DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20001	G_ENABLE_DV_ENFORCE_AUDIT_CODE
20002	G_DISABLE_DV_ENFORCE_AUDIT_CODE
20003	G_REALM_CREATION_AUDIT_CODE
20004	G_REALM_UPDATE_AUDIT_CODE
20005	G_REALM_RENAME_AUDIT_CODE
20006	G_REALM_DELETION_AUDIT_CODE
20007	G_ADD_REALM_AUTH_AUDIT_CODE
20008	G_DELETE_REALM_AUTH_AUDIT_CODE

Table 22–2 (Cont.) DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20009	G_UPDATE_REALM_AUTH_AUDIT_CODE
20010	G_ADD_REALM_OBJECT_AUDIT_CODE
20011	G_UPDATE_REALM_OBJECT_AUDIT_CODE
20012	G_DELETE_REALM_OBJECT_AUDIT_CODE
20013	G_ENABLE_EVENT_AUDIT_CODE
20014	G_DISABLE_EVENT_AUDIT_CODE
20015	G_RULE_SET_CREATION_AUDIT_CODE
20016	G_RULE_SET_UPDATE_AUDIT_CODE
20017	G_RULE_SET_RENAME_AUDIT_CODE
20018	G_RULE_SET_DELETION_AUDIT_CODE
20019	G_ADD_RULE_TO_RULE_SET_AUDIT_CODE
20020	G_DELETE_RULE_FROM_RULE_SET_AUDIT_CODE
20021	G_RULE_CREATION_AUDIT_CODE
20022	G_RULE_UPDATE_AUDIT_CODE
20023	G_RULE_RENAME_AUDIT_CODE
20024	G_RULE_DELETION_AUDIT_CODE
20025	G_COMMAND_RULE_CREATION_AUDIT_CODE
20026	G_COMMAND_RULE_UPDATE_AUDIT_CODE
20027	G_COMMAND_RULE_DELETION_AUDIT_CODE
20028	G_AUTHORIZE_DATAPUMP_USER_AUDIT_CODE
20029	G_UNAUTHORIZE_DATAPUMP_USER_AUDIT_CODE
20030	G_AUTHORIZE_JOB_USER_AUDIT_CODE
20031	G_UNAUTHORIZE_JOB_USER_AUDIT_CODE
20032	G_FACTOR_TYPE_CREATION_AUDIT_CODE
20033	G_FACTOR_TYPE_DELETION_AUDIT_CODE
20034	G_FACTOR_TYPE_UPDATE_AUDIT_CODE
20035	G_FACTOR_TYPE_RENAME_AUDIT_CODE
20036	G_FACTOR_CREATION_AUDIT_CODE
20037	G_FACTOR_DELETION_AUDIT_CODE
20038	G_FACTOR_UPDATE_AUDIT_CODE
20039	G_FACTOR_RENAME_AUDIT_CODE
20040	G_ADD_FACTOR_LINK_AUDIT_CODE
20041	G_DELETE_FACTOR_LINK_AUDIT_CODE
20042	G_ADD_POLICY_FACTOR_AUDIT_CODE
20043	G_DELETE_POLICY_FACTOR_AUDIT_CODE
20044	G_IDENTITY_CREATION_AUDIT_CODE
20045	G_IDENTITY_DELETION_AUDIT_CODE

Table 22–2 (Cont.) DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20046	G_IDENTITY_UPDATE_AUDIT_CODE
20047	G_CHANGE_IDENTITY_FACTOR_AUDIT_CODE
20048	G_CHANGE_IDENTITY_VALUE_AUDIT_CODE
20049	G_IDENTITY_MAP_CREATION_AUDIT_CODE
20050	G_IDENTITY_MAP_DELETION_AUDIT_CODE
20051	G_POLICY_LABEL_CREATION_AUDIT_CODE
20052	G_POLICY_LABEL_DELETION_AUDIT_CODE
20053	G_MAC_POLICY_CREATION_AUDIT_CODE
20054	G_MAC_POLICY_UPDATE_AUDIT_CODE
20055	G_MAC_POLICY_DELETION_AUDIT_CODE
20056	G_ROLE_CREATION_AUDIT_CODE
20057	G_ROLE_DELETION_AUDIT_CODE
20058	G_ROLE_UPDATE_AUDIT_CODE
20059	G_ROLE_RENAME_AUDIT_CODE
20060	G_DOMAIN_IDENTITY_CREATION_AUDIT_CODE
20061	G_DOMAIN_IDENTITY_DROP_AUDIT_CODE
20062	G_ORADEBUG_ENABLE_AUDIT_CODE
20063	G_ORADEBUG_DISABLE_AUDIT_CODE
20064	G_AUTHORIZE_PROXY_USER_AUDIT_CODE
20065	G_UNAUTHORIZE_PROXY_USER_AUDIT_CODE
20066	G_DV_DICT_ACCTS_ENABLE_AUDIT_CODE
20067	G_DV_DICT_ACCTS_DISABLE_AUDIT_CODE
20068	G_AUTHORIZE_DDL_AUDIT_CODE
20069	G_UNAUTHORIZE_DDL_AUDIT_CODE
20070	G_AUTHORIZE_TTS_AUDIT_CODE
20071	G_UNAUTHORIZE_TTS_AUDIT_CODE

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

- SYS.DV$ENFORCEMENT_AUDIT View

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR(128)		Operating system login user name of the user whose actions were audited
USERNAME	VARCHAR(128)		Name of the database user whose actions were audited
USERHOST	VARCHAR(255)		Client computer name
TERMINAL	VARCHAR(255)		Identifier for the user's terminal
TIMESTAMP	DATE		Date and time of creation of the audit trail entry (in the local database session time zone)
OWNER	VARCHAR(128)		Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR(128)		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 22–2 on page 22-23 for a listing of the possible actions.
ACTION_NAME	VARCHAR(128)		Name of the action type corresponding to the numeric code in the ACTION column
ACTION_OBJECT_ID	NUMBER		The unique identifier of the record in the table specified under OBJ_NAME
ACTION_OBJECT_NAME	VARCHAR(128)		The unique name or natural key of the record in the table specified under OBJ_NAME
ACTION_COMMAND	VARCHAR2(4000)		The SQL text of the command procedure that was executed that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)		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			The unique identifier of the rule set that was executing and caused the audit event to trigger
RULE_SET_NAME	VARCHAR(128)		The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER		Not used
RULE_NAME	VARCHAR2(128)		Not used

Column	Datatype	Null	Description
FACTOR_CONTEXT	VARCHAR2(4000)		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)		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		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		Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)		Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER		Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)		Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(128)		Database login user name of the user whose actions were audited
CREATE_DATE	DATE		Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(128)		Same as CREATED_BY column value
UPDATE_DATE	DATE		Same as UPDATED_BY column value

SYS.DV$CONFIGURATION_AUDIT View

The SYS.DV$CONFIGURATION_AUDIT view is almost the same as the DVSYS.DV$CONFIGURATION_AUDIT view except that it captures Database Vault-related audit records from the unified audit trail.

See Also: "DVSYS.DV$CONFIGURATION_AUDIT View" on page 22-21

SYS.DV$ENFORCEMENT_AUDIT View

The SYS.DV$ENFORCEMENT_AUDIT view is almost the same as the DVSYS.DV$ENFORCEMENT_AUDIT view except that it captures Database Vault-related audit records from the unified audit trail.

See Also: "DVSYS.DV$ENFORCEMENT_AUDIT View" on page 22-25

23 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.

Topics:

- Monitoring Security Violation Attempts
- Monitoring Security Policy Changes

Monitoring Security Violation Attempts

You should periodically check for violations to the Oracle Database Vault configuration.

Topics:

- About Monitoring Security Violation Attempts
- Using Oracle Database Vault Administrator to Monitor Security Violations

About Monitoring Security Violation Attempts

You can check for security violations, such as realm or command rule violations. This feature displays data such as the user name of the person committing the violation, the action they committed, and a time stamp of the activity.

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.

Using Oracle Database Vault Administrator to Monitor Security Violations

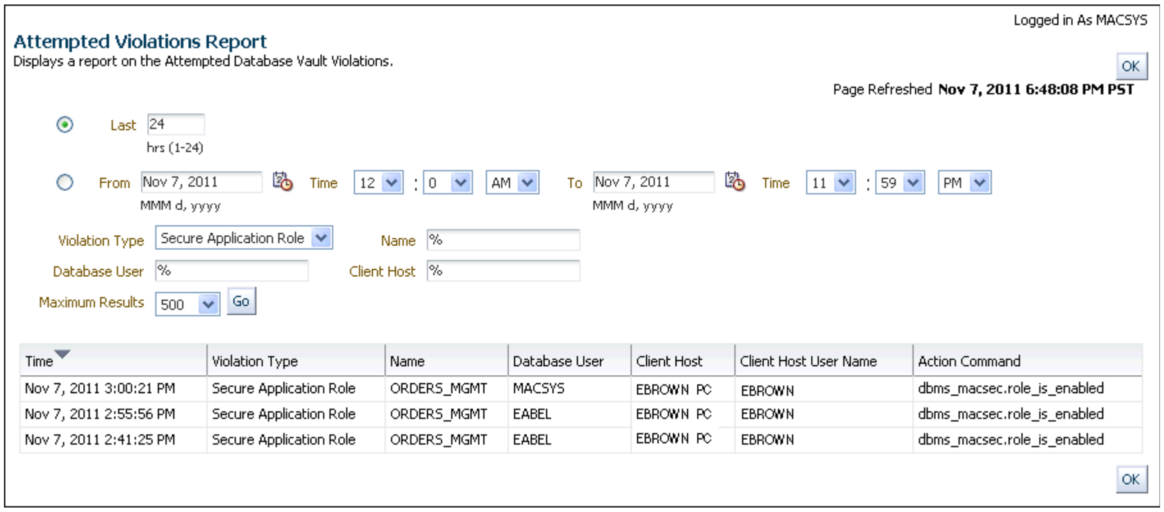
A user who has been granted the appropriate role can use Oracle Database Vault Administrator to monitor security violations.

1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, `DV_MONITOR`, or `DV_SECANALYST` role.

   "Logging into Oracle Database Vault" on page 3-7 explains how to log in.

2. In the Home page, under Reports, select **Attempted Violations**.
3. In the Attempted Violations Report page, set the period of time and other filter settings to define the data that you want to capture, and then click **Go**.

The report appears, similar to the following page:


Logged in As MACSYS

Attempted Violations Report

Displays a report on the Attempted Database Vault Violations.

Page Refreshed **Nov 7, 2011 6:48:08 PM PST**

[✓ Last] 24 hrs (1-24)

From Nov 7, 2011 MMM d, yyyy Time 12 : 0 AM To Nov 7, 2011 MMM d, yyyy Time 11 : 59 PM

Violation Type: Secure Application Role

Name: %

Database User: %

Client Host: %

Maximum Results: 500

Time	Violation Type	Name	Database User	Client Host	Client Host User Name	Action Command
Nov 7, 2011 3:00:21 PM	Secure Application Role	ORDERS_MGMT	MACSYS	EBROWN PC	EBROWN	dbms_macsec.role_is_enabled
Nov 7, 2011 2:55:56 PM	Secure Application Role	ORDERS_MGMT	EABEL	EBROWN PC	EBROWN	dbms_macsec.role_is_enabled
Nov 7, 2011 2:41:25 PM	Secure Application Role	ORDERS_MGMT	EABEL	EBROWN PC	EBROWN	dbms_macsec.role_is_enabled

4. To exit, click **OK**.

Monitoring Security Policy Changes

You should periodically check for changes to the Oracle Database Vault configuration.

Topics:

- About Monitoring Security Policy Changes
- Using Oracle Database Vault to Monitor Security Policy Changes by Category

About Monitoring Security Policy Changes

The Database Vault Policy Change Report in Oracle Database Vault Administrator enables you to track changes that have been made to security settings.

You can check the number of policy changes for the categories in the following list. These categories reflect changes to the database security policy (that is, its configuration) in any given environment. If something changes that is security related, you can use the chart and tables to drill down to find unexpected changes that should be investigated.

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, you do not need to configure any additional settings. You are ready to check for changes to Database Vault policies.

- **Database Vault policy:** Shows changes made through the Oracle Database Vault administrative packages or user interface, indicating Oracle Database Vault configuration or policy changes.
- **Label Security policy:** Shows changes made through the Oracle Database Vault administrative packages or user interface, indicating Oracle Label Security policy or privilege changes.
- **Audit Policy:** Shows changes to the database audit policy coming from `AUDIT` or `NOAUDIT` statements.
- **Privilege Grants:** Shows changes to system or object privilege `GRANT` statements.

- **Privilege Revokes:** Shows changes to system or object privilege `REVOKE` statements.
- **Database Account:** Shows changes to `CREATE USER`, `ALTER USER`, or `DROP USER` statements.
- **Database Role:** Shows changes to `CREATE ROLE`, `ALTER ROLE`, or `DROP ROLE` statements.

Using Oracle Database Vault to Monitor Security Policy Changes by Category

A user who has been granted the appropriate role can use Oracle Database Vault Administrator to monitor security policy changes by category.

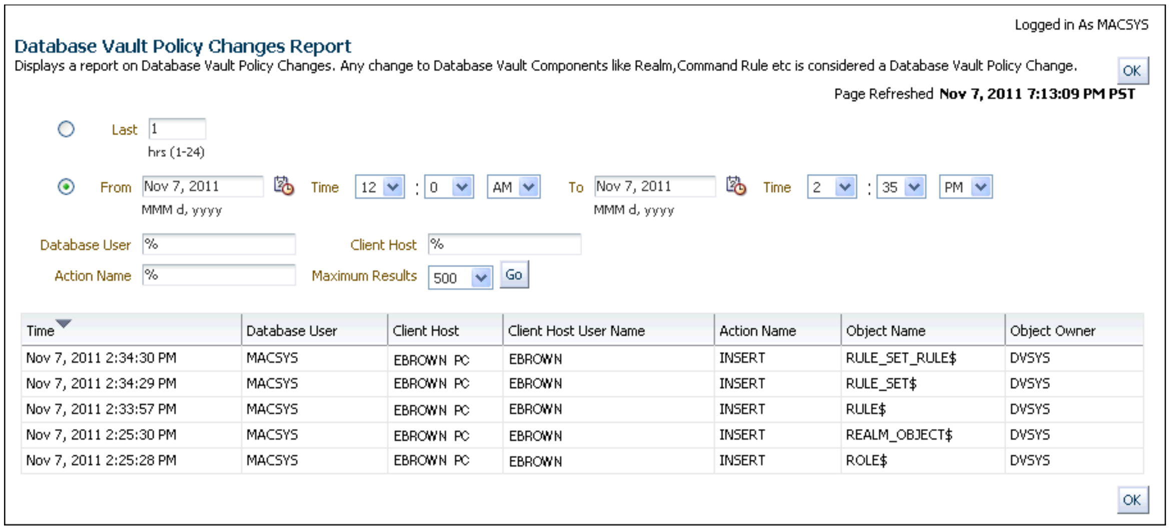
1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, `DV_MONITOR`, or `DV_SECANALYST` role.

   "Logging into Oracle Database Vault" on page 3-7 explains how to log on.

2. In the Home page, under Reports, select **Database Vault Policy Changes**.

3. In the Attempted Violations Report page, set the period of time and other filter settings to define the data that you want to capture, and then click **Go**.

   The report appears, similar to the following page:



Database Vault Policy Changes Report

Displays a report on Database Vault Policy Changes. Any change to Database Vault Components like Realm,Command Rule etc is considered a Database Vault Policy Change.

Page Refreshed **Nov 7, 2011 7:13:09 PM PST**

Last: 1 hrs (1-24)

[✓ From] Nov 7, 2011 (MMM d, yyyy) Time 12 : 0 AM To Nov 7, 2011 (MMM d, yyyy) Time 2 : 35 PM

Database User: % Client Host: %

Action Name: % Maximum Results: 500

Time	Database User	Client Host	Client Host User Name	Action Name	Object Name	Object Owner
Nov 7, 2011 2:34:30 PM	MACSYS	EBROWN PC	EBROWN	INSERT	RULE_SET_RULE$	DVSYS
Nov 7, 2011 2:34:29 PM	MACSYS	EBROWN PC	EBROWN	INSERT	RULE_SET$	DVSYS
Nov 7, 2011 2:33:57 PM	MACSYS	EBROWN PC	EBROWN	INSERT	RULE$	DVSYS
Nov 7, 2011 2:25:30 PM	MACSYS	EBROWN PC	EBROWN	INSERT	REALM_OBJECT$	DVSYS
Nov 7, 2011 2:25:28 PM	MACSYS	EBROWN PC	EBROWN	INSERT	ROLE$	DVSYS

4. To exit, click **OK**.

24 Oracle Database Vault Reports

Oracle Database Vault provides a set of reports that you can use to track activities, such as the Database Vault configuration settings.

Topics:

See Also:

- "Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control" on page 11-4
- Chapter 22, "Oracle Database Vault Data Dictionary Views" for additional ways that you can track Database Vault activities

About the Oracle Database Vault Reports

Oracle Database Vault provides a selection of 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.

Who Can Run the Oracle Database Vault Reports?

You must log on using an account that has the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role before you can run the Oracle Database Vault reports.

See Also:

- "DV_OWNER Database Vault Owner Role" on page 12-6
- "DV_ADMIN Database Vault Configuration Administrator Role" on page 12-8
- "DV_SECANALYST Database Vault Security Analyst Role" on page 12-10

Running the Oracle Database Vault Reports

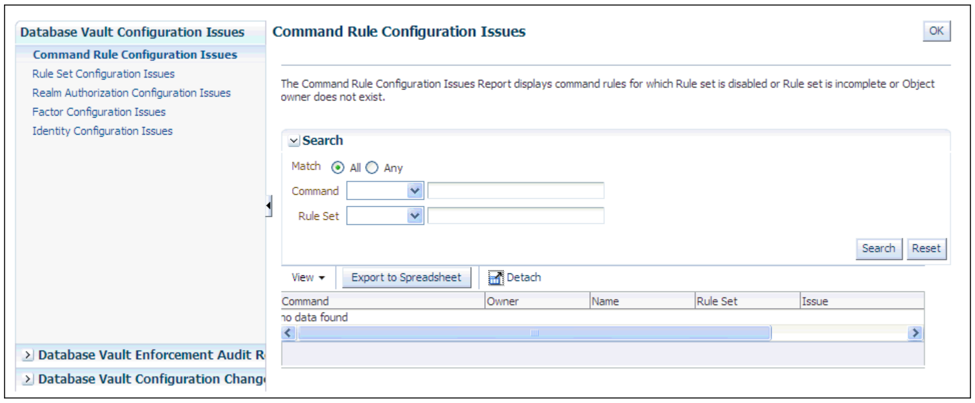
A user who has been granted the appropriate roles can run the Oracle Database Vault reports from Database Vault Administrator.

1. From Cloud Control, log into Database Vault Administrator as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

   "Logging into Oracle Database Vault" on page 3-7 explains how to log in.

2. In the Home page, under Reports, select **Database Vault Reports**.

   A page similar to the following appears:



Database Vault Configuration Issues
Command Rule Configuration Issues
Rule Set Configuration Issues
Realm Authorization Configuration Issues
Factor Configuration Issues
Identity Configuration Issues
Database Vault Enforcement Audit R
Database Vault Configuration Chang

Command Rule Configuration Issues

The Command Rule Configuration Issues Report displays command rules for which Rule set is disabled or Rule set is incomplete or Object owner does not exist.

Search
Match [✓ All] Any
Command
Rule Set

Command	Owner	Name	Rule Set	Issue
no data found				

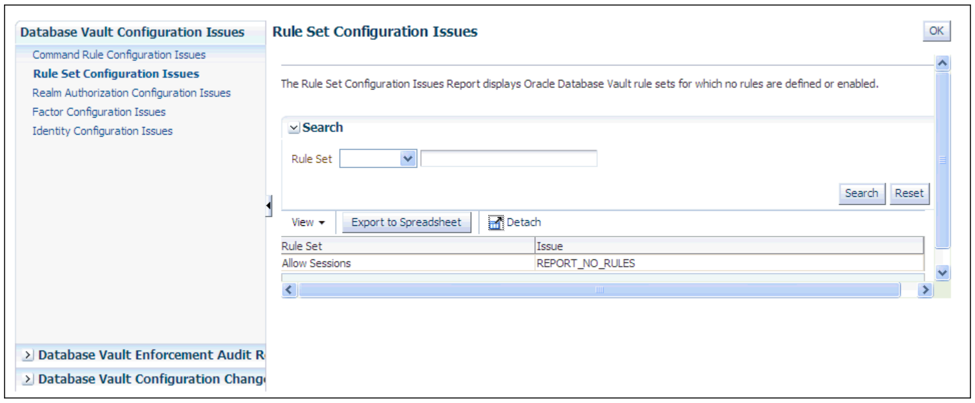
3. On the left side, select the category of reports that you want.
   - Database Vault Configuration Issues
   - Database Vault Enforcement Audit Reports
   - Database Vault Configuration Changes

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.


Database Vault Configuration Issues
Command Rule Configuration Issues
Rule Set Configuration Issues
Realm Authorization Configuration Issues
Factor Configuration Issues
Identity Configuration Issues
Database Vault Enforcement Audit R
Database Vault Configuration Chang
Rule Set Configuration Issues
OK
Search
Rule Set
Search
Reset
View
Export to Spreadsheet
Detach

The Rule Set Configuration Issues Report displays Oracle Database Vault rule sets for which no rules are defined or enabled.

Rule Set	Issue
Allow Sessions	REPORT_NO_RULES

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 that have been for command rules, rule sets, realms, and other Oracle Database Vault configurations.

Topics:

- Command Rule Configuration Issues Report
- Rule Set Configuration Issues Report
- Realm Authorization Configuration Issues Report
- Factor Configuration Issues Report
- Identity Configuration Issues Report
- Factor Without Identities Report
- Secure Application Configuration Issues Report

Command Rule Configuration Issues Report

The Command Rule Configuration Issues Report displays command rules for which specific configuration issues exist.

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 information.

It 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 information where the certain configuration issues exist.

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 for which the specific configuration issues exist.

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 defined in the access control configuration.

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 identities where specific configuration issues exist.

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 information where specific configuration issues exist.

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

You can track auditing issues with the Oracle Database Vault auditing reports. Remember that if you have unified auditing enabled, then the reports capture the results of your unified audit policies.

Topics:

- Realm Audit Report
- Command Rule Audit Report
- Factor Audit Report
- Label Security Integration Audit Report
- Core Database Vault Audit Trail Report
- Secure Application Role Audit Report

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. It 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 generated by the session initialization operation 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 generated by the core access security session initialization operation.

You can audit instances where the access security session fails to initialize. It displays the following data:

Violation Attempt	Instance Number
Timestamp	Object Name
Return Code	Rule Set
Account	Command
User Host	

Secure Application Role Audit Report

The Secure Application Role Audit Report shows the audit records generated by the secure application role-enabling operation for Oracle Database Vault.

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.

Topics:

- Object Privilege Reports
- Database Account System Privileges Reports
- Sensitive Objects Reports
- Privilege Management - Summary Reports
- Powerful Database Accounts and Roles Reports
- Initialization Parameters and Profiles Reports
- Database Account Password Reports
- Security Audit Report: Core Database Audit Report
- Other Security Vulnerability Reports

Object Privilege Reports

The object privilege reports track privileges affected by `PUBLIC`, direct object privileges, and object dependencies.

Topics:

- Object Access By PUBLIC Report
- Object Access Not By PUBLIC Report
- Direct Object Privileges Report
- Object Dependencies Report

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 all the object access the database accounts that you specify on the Report Parameters page, through 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:

`CTXSYS`	`LBACSYS`	`PUBLIC`	`SYSTEM`
`DMSYS`	`MDSYS`	`SYS`	`WKSYS`
`DVSYS`	`ORDSYS`	`SYSMAN`	`WMSYS`

Object Dependencies Report

The Object Dependencies Report describes all dependencies in the database between procedures, packages, functions, package bodies, and triggers, including 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.

Topics:

- Direct System Privileges By Database Account Report
- Direct and Indirect System Privileges By Database Account Report
- Hierarchical System Privileges by Database Account Report
- ANY System Privileges for Database Accounts Report
- System Privileges By Privilege Report

Direct System Privileges By Database Account Report

The Direct System Privileges By Database Account Report displays all system privileges that have been 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 all the 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 displays 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 all `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 displays the database accounts and roles that have the system privilege selected on the Report Parameters page.

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.

Topics:

- Execute Privileges to Strong SYS Packages Report
- Access to Sensitive Objects Report

- Public Execute Privilege To SYS PL/SQL Procedures Report
- Accounts with SYSDBA/SYSOPER Privilege Report

Execute Privileges to Strong SYS Packages Report

The Execute Privileges to Strong SYS Packages Report shows the database accounts and roles that have the `EXECUTE` privilege on system packages that can be used to access operating system resources or other powerful system packages.

The following system PL/SQL packages are included:

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 the database accounts and roles that have object privileges on system tables or views that contain sensitive information.

This report includes the following system tables and views:

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
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 all database accounts and roles that have execute privileges on packages owned by `SYS`.

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.

Topics:

- Privileges Distribution By Grantee Report
- Privileges Distribution By Grantee, Owner Report
- Privileges Distribution By Grantee, Owner, Privilege Report

See Also: "DVSYS.DBA_DV_PUB_PRIVS View" on page 22-13 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 owner of the object.

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`, `BECOME USER`, `ALTER SYSTEM`, `ALTER SESSION`, `WITH GRANT`, and `AUDIT` privileges.

Topics:

- WITH ADMIN Privilege Grants Report
- Accounts With DBA Roles Report
- Security Policy Exemption Report
- BECOME USER Report
- ALTER SYSTEM or ALTER SESSION Report
- Password History Access Report
- WITH GRANT Privileges Report
- Roles/Accounts That Have a Given Role Report
- Database Accounts With Catalog Roles Report
- AUDIT Privileges Report
- OS Security Vulnerability Privileges Report

See Also:

- "DVSYS.DBA_DV_PUB_PRIVS View" on page 22-13
- "DVSYS.DBA_DV_USER_PRIVS View" on page 22-20
- "DVSYS.DBA_DV_USER_PRIVS_ALL View" on page 22-21

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.

For guidelines on deciding who should have privileged roles, see Appendix D, "Oracle Database Vault Security Guidelines."

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 granted to them.

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 Appendix A, "Auditing Oracle Database Vault," to audit the use of this privilege.

BECOME USER Report

The BECOME USER Report shows all 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 all 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.

For guidelines on using the `ALTER SYSTEM` and `ALTER SESSION` privileges, see "Security Considerations for the ALTER SYSTEM and ALTER SESSION Privileges" on page D-12.

Password History Access Report

The Password History Access Report shows database accounts that have access to the `USER_HISTORY$` table that 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 all database accounts that have been 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 shows the database accounts and roles that have the required system privileges to export sensitive or otherwise protected 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.

Topics:

- Security Related Database Parameters Report
- Resource Profiles Report
- System Resource Limits Report

Security Related Database Parameters Report

The Security Related Database Parameters Report displays database parameters that can cause security vulnerabilities, if 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 provides a view of resource profiles, such as CPU_PER_SESSION and IDLE_TIME, that may be allowing unlimited resource consumption.

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.

Topics:

- Database Account Default Password Report
- Database Account Status Report

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 provides a quick view of 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 returns audit records for the audit policy defined in Appendix A, "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. For more information about the AUDIT_TRAIL parameter, see *Oracle Database SQL Language Reference*.

Other Security Vulnerability Reports

The other security vulnerability reports track issues with vulnerabilities that can arise with activities just Java policy grants, operating system directory objects, unwrapped PL/SQL packages, and non-owner object triggers.

Topics:

- Java Policy Grants Report
- OS Directory Objects Report
- Objects Dependent on Dynamic SQL Report
- Unwrapped PL/SQL Package Bodies Report
- Username/Password Tables Report
- Tablespace Quotas Report
- Non-Owner Object Trigger Report

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 all directory objects that exist in the database, whether they are available to `PUBLIC`, and what their privileges are.

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 shows objects that leverage 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 displays 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 helps to identify 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 shows all 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 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, by using the Database Vault APIs or unified auditing PL/SQL statements.

Topics:

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 Chapter 24, "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 PL/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, SYS.DV$CONFIGURATION_AUDIT, and SYS.DV$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" on page A-5. 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
- "Format of the Oracle Database Vault DVSYS.AUDIT_TRAIL$ Audit Trail Records" on page A-3
- The following data dictionary views, which are specific to Database Vault unified auditing:

  "DVSYS.DV$CONFIGURATION_AUDIT View" on page 22-21

  "DVSYS.DV$ENFORCEMENT_AUDIT View" on page 22-25
- *Oracle Database Upgrade Guide* to migrate your database to unified auditing

Protecting 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.

A realm is not necessary for the AUDSYS schema because the information it contains is internal and cannot be viewed in the same way that other database objects (such as tables or views) can be viewed in a schema. To protect the unified audit trail, create a realm around the AUDIT_ADMIN and AUDIT_VIEWER roles. This way, you can control who can grant these roles.

See Also: "Creating a Realm" on page 5-6

Oracle Database Vault Specific Audit Events

Oracle Database Vault audit events track activities such as whether an action attempted on a realm was successful or not.

Topics:

- Oracle Database Vault Policy Audit Events
- Format of the Oracle Database Vault DVSYS.AUDIT_TRAIL$ Audit Trail Records

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.

See Also:

- "Setting Audit Options for a Factor" on page 8-11 (for factors)
- "About Realm Authorization" on page 5-9
- Chapter 24, "Oracle Database Vault Reports," for information about viewing the audit reports

Format of the Oracle Database Vault DVSYS.AUDIT_TRAIL$ Audit Trail Records

If you choose not to use unified auditing, then Oracle Database Vault will write audit records to the `AUDIT_TRAIL$` table, which is in the `DVSYS` schema.

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 `AUDIT_TRAIL$` table. In fact, even if auditing has been disabled in Oracle Database, then the Oracle Database Vault audit functionality continues to write to the `AUDIT_TRAIL$` 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 `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)`		Operating system login user name of the user whose actions were audited
`USERNAME`	`VARCHAR2(30)`		Name of the database user whose actions were audited
`USERHOST`	`VARCHAR2(128)`		Client computer name
`TERMINAL`	`VARCHAR2(255)`		Identifier for the user's terminal
`TIMESTAMP`	`DATE`		Date and time of creation of the audit trail entry (in the local database session time zone)

Table A–1 (Cont.) Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
OWNER	VARCHAR2(30)		Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR2(128)		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 22–2, " DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values" on page 22-23 for a list of the expected ACTION and ACTION_NAME values.
ACTION_NAME	VARCHAR2(128)		Name of the action type corresponding to the numeric code in the ACTION column
ACTION_OBJECT_ID	NUMBER		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)		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)		The SQL text of the command procedure that was executed that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)		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		The unique identifier of the rule set that was executing and caused the audit event to trigger
RULE_SET_NAME	VARCHAR2(30)		The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER		Not used
RULE_NAME	VARCHAR2(30)		Not used
FACTOR_CONTEXT	VARCHAR2(4000)		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)		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.

Table A–1 (Cont.) Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
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		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		Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)		Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER		Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)		Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(30)		Database login user name of the user whose actions were audited
CREATE_DATE	DATE		Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(30)		Same as CREATED_BY column value
UPDATE_DATE	DATE		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.

Topics:

- About Archiving and Purging the Oracle Database Vault Audit Trail
- Archiving the Oracle Database Vault Audit Trail
- Purging the Oracle Database Vault Audit Trail

About Archiving and Purging the Oracle Database Vault Audit Trail

In a non-unified auditing environment, you can create an archive of the Oracle Database Vault audit trail by exporting the AUDIT_TRAIL$ system table, which is owned by DVSYS, 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, SYS.DV$CONFIGURATION_AUDIT, and SYS.DV$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" on page 11-4 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
    - Chapter 13, "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$;
```

See Also: "DV_AUDIT_CLEANUP Audit Trail Cleanup Role" on page 12-10

Oracle Database Audit Settings Created for Oracle Database Vault

When you install Oracle Database Vault, it creates several `AUDIT` settings in the database.

However, 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)
System Audit Settings/System Privilege Usage	`ALTER ANY`
	`CREATE ANY`
	`DELETE ANY`
	`DROP ANY`
	`EXECUTE ANY` (`WHENEVER NOT SUCCESSFUL`)
	`FORCE ANY`
	`GRANT ANY`
	`INSERT ANY`
	`UPDATE ANY`
System Audit Settings/Object Management	`ALTER DATABASE, PROFILE, ROLLBACK SEGMENT, SESSION, SYSTEM, TABLE, TABLESPACE, USER`
	`CREATE CLUSTER, DATABASE LINK, INDEXTYPE, LIBRARY, OPERATOR, PUBLIC SYNONYM, PROCEDURE, PROFILE, ROLE, ROLLBACK SEGMENT, SEQUENCE, SESSION, SNAPSHOT, SYNONYM, TABLE, TABLESPACE, TRIGGER, TYPE, USER, VIEW`
	`TRUNCATE`
System Audit Settings/Intrusive Commands	`ALTER SESSION`
	`BECOME USER`
	`CREATE SESSION`
	`DEBUG CONNECT SESSION`
	`RESTRICTED SESSION`

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)
System Audit Settings/Administration Commands	`ADMINISTER DATABASE TRIGGER`
	`BACKUP ANY TABLE`
	`EXEMPT ACCESS POLICY`
	`MANAGE TABLESPACE`
System Audit Settings/Audit Commands	`AUDIT ANY`
	`AUDIT SYSTEM`
System Audit Settings/Access Control	`GRANT ANY PRIVILEGE/ANY OBJECT PRIVILEGE/ROLE`
	`GRANT DIRECTORY`
	`GRANT SEQUENCE`
	`GRANT TABLE`
	`GRANT TYPE`

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)
User Audit Settings for `DVSYS/DVF`	`ADMINISTER DATABASE TRIGGER`
User Audit Settings for `LBACSYS`	`ALTER` *`object`*
See Table 12–2, " Database Accounts Used by Oracle Database Vault" on page 12-20 for more information about these accounts.	`AUDIT SYSTEM`
	`BECOME USER`
	`CLUSTER`
See also these sections for detailed information on the `DVSYS` and `DVF` schemas:	`COMMENT`
	`CONTEXT`
■ "DVSYS Schema" on page 12-1	`CREATE` *`object`*
■ "DVF Schema" on page 12-2	`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`)
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 Oracle Database Vault is disabled, there are some Database Vault features that you can still use.

Topics:

Note: Oracle does not support the deinstallation of Oracle Database Vault.

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.

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" on page 2-4 for a listing of the revoked privileges.

The following situations require you to disable Oracle Database Vault:

- The Oracle Database Vault user accounts have been 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 the tip under "Oracle Database Vault Accounts" on page 12-19 for a guideline for avoiding this problem in the future.)

- 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).

Checking if Oracle Database Vault Is Enabled or Disabled

You can check if Oracle Database Vault is enabled or disabled by querying the `V$OPTION` data dictionary view. Any user can query this view.

If Oracle Database Vault is enabled, the query returns `TRUE`. Otherwise, it returns `FALSE`.

Remember that the `PARAMETER` column value is case sensitive. For example:

```
SELECT PARAMETER, VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';
```

If Oracle Database Vault is enabled, the following output appears:

```
PARAMETER                    VALUE
---------------------------- -----------------------
Oracle Database Vault        TRUE
```

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.

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" on page 12-19 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 Oracle Database Vault.

```
CONNECT psmith -- Or, CONNECT psmith@hrpdb for a PDB
Enter password: password

EXEC DBMS_MACADM.ENABLE_DV;
```

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 have registered Oracle Database Vault, you can configure it on Oracle Real Application Clusters (Oracle RAC) nodes and add languages, if necessary. You can also de-install and reinstall Oracle Database Vault.

Topics:

- Configuring Oracle Database Vault on Oracle RAC Nodes
- Adding Languages to Oracle Database Vault
- Deinstalling Oracle Database Vault
- Reinstalling Oracle Database Vault

> **See Also:** "Plugging a Database Vault-Enabled PDB to a CDB" on page 11-18 if you are using a multitenant container database (CDB)

Configuring Oracle Database Vault on Oracle RAC Nodes

After you enable and register Oracle Database Vault for an Oracle Real Application Clusters (Oracle RAC) instance, complete the following procedure for each Oracle RAC node. This 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. 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. Register Database Vault.

   See "Registering Oracle Database Vault with an Oracle Database" on page 3-1.

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.

Topics:

- Separation of Duty Guidelines
- Managing Oracle Database Administrative Accounts
- Accounts and Roles Trusted by Oracle Database Vault
- Accounts and Roles That Should be Limited to Trusted Individuals
- Guidelines for Using Oracle Database Vault in a Production Environment
- Secure Configuration Guidelines

Separation of Duty Guidelines

Oracle Database Vault is designed to easily implement separation of duty guidelines.

Topics:

- How Oracle Database Vault Handles Separation of Duty
- Separation of Tasks in an Oracle Database Vault Environment
- Separation of Duty Matrix for Oracle Database Vault
- Identification and Documentation of the Tasks of Users Who Access the Database System

How Oracle Database Vault Handles Separation of Duty

Separation of duty means that you restrict 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.
- **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. (For an in-depth look at how the Oracle Database Vault roles provide for separation of duty, see "Oracle Database Vault Roles" on page 12-2.)

  Optionally, you can consolidate the account management and security administrative responsibilities.
- **Resource management.** Resource 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

  For resource management, you should create a named account and a backup account for each of these tasks. Use these accounts as the primary resource managers in the database.

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.

Separation of Duty Matrix for Oracle Database Vault

Before separation of duty can be successful, 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. 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`	X						
`SHARDY`							X
`PKESTNER`			X				
`RTYLER`					X		
`SANDERSON`				X		X	
`SYSTEM`					EBS patching		
`RMAN`		X	X				
...							

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 Users Who Access the Database System

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

You should be aware of how best to manage the security of the administrative accounts, such as SYSTEM, users who have the SYSDBA administrative privilege, and the root and operating system access by users.

Topics:

- SYSTEM User Account for General Administrative Uses
- SYSTEM Schema for Application Tables
- Limitation of the SYSDBA Administrative Privilege
- Root and Operating System Access to Oracle Database Vault

SYSTEM User Account for General Administrative Uses

If you use the SYSTEM account for general database administrative purposes, create named database administrative accounts for your database administrators.

Doing so increases accountability for administrative actions in the database.

SYSTEM Schema for Application Tables

If your site holds application tables in the SYSTEM schema, then you should add the SYSTEM account to your realm authorizations for these tables so that these applications can 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

You should limit the SYSDBA administrative privilege only to users who must connect using this privilege when absolutely necessary and for applications that still require SYSDBA access, such as mandatory patching processes. 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.

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 hide individual table columns. (See *Oracle Database Advanced Security Guide* for more information about Transparent Data Encryption.) 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.

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 installation and used for creating new database accounts
DV_OWNER role	Open	Role created during installation and used for managing realms, factors and command rules. This user can add himself or herself to realm authorizations. Users who have the DV_ACCTMGR role cannot alter this user.
SYSDBA privilege	Enabled	Privilege created during Oracle Database installation. Required by some Oracle features. See "Management of SYSDBA Access" on page D-6 for guidelines on managing SYSDBA.
SYSOPER privilege	Enabled	Privilege created during Oracle Database installation. Database startup and shutdown. Granted to SYS only by default. See "Management of SYSOPER Access" on page D-6 for guidelines on managing SYSOPER.

Accounts and Roles That Should be Limited to Trusted Individuals

Several accounts and roles have very powerful privileges in a default Oracle Database installation. You should limit these accounts and roles only to trusted individuals.

- Management of Users with Root Access to the Operating System
- Management of the Oracle Software Owner
- Management of SYSDBA Access
- Management of SYSOPER Access

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. Ensure that you grant root user privileges only to the appropriate people with the appropriate responsibility.

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. Ensure that you grant Oracle software owner access only to the appropriate people with the appropriate responsibility.

Management of SYSDBA Access

By default, Oracle Database limits `SYSDBA` access to administrative users in the `OSDBA` group and to the user `SYS`.

The `SYSDBA` administrative privilege is a trusted privilege in Oracle Database Vault. Grant this privilege to trusted users only.

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.

- Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages
- Security Considerations for the Recycle Bin
- Security Considerations for the CREATE ANY JOB Privilege
- Security Considerations for the CREATE EXTERNAL JOB Privilege
- Security Considerations for the LogMiner Packages
- Security Considerations for the ALTER SYSTEM and ALTER SESSION Privileges

Note:

- 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 `SYSTEM` 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.

See Also:

- "Privileges That Are Revoked from Existing Users and Roles" on page 2-4
- Table 12–1, " Privileges of Oracle Database Vault Roles" on page 12-5

Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages

You should carefully restrict access to the UTL_FILE and DBMS_FILE_TRANSFER PL/SQL packages.

Topics:

- About Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages
- Securing Access to the DBMS_FILE_TRANFER Package
- 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

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.

You can configure the `UTL_FILE` package securely; see *Oracle Database PL/SQL Packages and Types Reference* for more information.

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.

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 or Editing a Command Rule" on page 7-4 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" on page D-9
    - "Example: Creating a Command Rule to Enable Access to CREATE DATABASE LINK" on page D-10
    - "Example: Command Rules to Disable and Enable Access to CREATE DIRECTORY" on page D-10

Example: Creating a Command Rule to Deny Access to CREATE DATABASE LINK

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

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;
```

Security Considerations for the Recycle Bin

You should be aware that realm-protected objects that are dropped are not protected in the recycle bin.

Topics:

- About Security Considerations for the Recycle Bin
- Checking the Contents of the Recycle Bin
- Purging the Contents of the Recycle Bin
- Disabling the Recycle Bin

About Security Considerations for the Recycle Bin

In Oracle Database Vault, the `ALTER SYSTEM` command rule prevents the recycle bin feature from being enabled, but if you enable the recycle bin, it cannot prevent you from disabling it.

When the recycle bin feature is enabled, realm-protected objects that are dropped go into the recycle bin. Once there, the objects are no longer protected by the realm. The exception is objects in the `DVSYS` schema: To keep `DVSYS` as a protected schema, you cannot drop its objects. For better security for other realms, you should disable the recycle bin.

Checking the Contents of the Recycle Bin

You can use the `SELECT` statement to check the contents of the recycle bin.

- In SQL*Plus, use `SELECT` as follows to check the recycle bin contents;

```
SELECT * FROM RECYCLEBIN;
SELECT * FROM USER_RECYCLEBIN;
```

Purging the Contents of the Recycle Bin

You can use the `PURGE` SQL statement to purge the contents of the recycle bin.

- Run the following `PURGE` commands to purge the recycle bin contents:

```
PURGE RECYCLEBIN;
PURGE USER_RECYCLEBIN;
```

Disabling the Recycle Bin

In an Oracle Database Vault environment, disabling the recycle bin prevents exposure to Database Vault objects that have dropped.

1. Log into the database instance as user `SYS` with the `SYSDBA` administrative privilege, or as user who has the `ALTER SYSTEM` system privilege.

```
sqlplus psmith
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. Restart Oracle Database.

```
CONNECT SYS AS SYSDBA
Enter password: password

SHUTDOWN IMMEDIATE
STARTUP
```

Security Considerations for the CREATE ANY JOB Privilege

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.

See "Using Oracle Scheduler with Oracle Database Vault" on page 11-12 for more information.

Security Considerations for the CREATE EXTERNAL JOB Privilege

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.

Security Considerations for the LogMiner Packages

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.

Security Considerations for the ALTER SYSTEM and ALTER SESSION Privileges

You should be aware of ways to secure the powerful `ALTER SYSTEM` and `ALTER SESSION` system privileges.

Topics:

- About Security Considerations for the ALTER SYSTEM and ALTER SESSION Privileges
- Example: Adding Rules to the Existing ALTER SYSTEM Command Rule

About Security Considerations for the ALTER SYSTEM and ALTER SESSION Privileges

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

Example D–4 shows how you can create a rule that prevents users with `ALTER SYSTEM` privilege from issuing the command `ALTER SYSTEM DUMP`. Log into the database instance as the Oracle Database Vault Owner when you create this command rule.

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(DVSYS.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;
```

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" on page 6-6 for more information.

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.

Topics:

- Using Trace Files to Diagnose Oracle Database Vault Events
- General Diagnostic Tips
- Configuration Problems with Oracle Database Vault Components

Using Trace Files to Diagnose Oracle Database Vault Events

Trace files are generated by the database. They can capture important information to help you debug errors.

Topics:

- About Using Trace Files to Diagnose Oracle Database Vault Events
- Types of Oracle Database Vault Trace Events That You Can and Cannot Track
- Levels of Oracle Database Vault Trace Events
- Performance Effect of Enabling Oracle Database Vault Trace Files
- Enabling Oracle Database Vault Trace Events
- Finding Oracle Database Vault Trace File Data
- Example: Low Level Oracle Database Vault Realm Violations in a Trace File
- Disabling Oracle Database Vault Trace Events

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

Table E–1 describes the types of activities that you can track with trace files.

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" on page E-6 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.
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.

Topics:

- Enabling Trace Events for the Current Database Session
- Enabling Trace Events for All Database Sessions
- Management of Trace Events 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" on page E-2.

   - 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" on page E-3.

   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

You can find Oracle Database vault trace file data by using the Linux grep command or by using the ADR Command Interpreter (`ADRCI`) command-line utility.

Topics:

- Finding the Database Vault Trace File Directory Location
- Using the Linux grep Command to Search Trace Files for Strings
- Using the ADR Command Interpreter (ADRCI) Utility to QueryTrace Files

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';

  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

Example E–1 shows trace file data for low level realm 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

Example E–2 shows how Database Vault authorization can appear in a trace file with high level trace enabled.

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

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<-opio
dr<-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.

Topics:

- Disabling Trace Events for the Current Database Session
- Disabling Trace Events for All Database Sessions
- Disabling Trace Events 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" on page E-9.

   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" on page 24-5 for more information.

Configuration Problems with Oracle Database Vault Components

If you suspect problems with the configuration of realms, command rules, factors, rule sets, or secure application roles, you can run the appropriate configuration report.

See the following sections for more information:

- "Command Rule Configuration Issues Report" on page 24-3
- "Factor Configuration Issues Report" on page 24-4
- "Factor Without Identities Report" on page 24-4
- "Identity Configuration Issues Report" on page 24-4
- "Realm Authorization Configuration Issues Report" on page 24-4
- "Rule Set Configuration Issues Report" on page 24-3

- "Secure Application Configuration Issues Report" on page 24-4

To run these reports, see "Running the Oracle Database Vault Reports" on page 24-2.

Index