Oracle® Database Vault

Administrator's Guide

11*g* Release 1 (11.1)

B31222-01

July 2007

ORACLE®

Oracle Database Vault Administrator’s Guide 11*g* Release 1 (11.1)

B31222-01

Primary Author: Patricia Huey

Contributors: Priya Badnar, Tom Best, Ji-won Byun, Ben Chang, Martin Cheng, Chi Ching Chui, Scott Gaetjen, Sumit Jeloka, Dominique Jeunot, Terri Keller, Frank Lee, Paul Needham, Deborah Owens, Vipin Samar, James Spiller, Ashwini Supur, Kamal Tbeileh, Peter Wahl, Daniel Wong, Aravind Yalamanchi

Contents

4 Configuring Realms

5 Configuring Rule Sets

6 Configuring Command Rules

10 Monitoring Oracle Database Vault

11 Oracle Database Vault Reports

A Oracle Database Vault Auditing Policies

B Enabling and Disabling Oracle Database Vault

C Post-Installation Oracle Database Vault Procedures

D Oracle Database Vault Objects

E Oracle Database Vault DVSYS.DBMS_MACADM Package

F Oracle Database Vault DVSYS.DBMS_MACSEC_ROLES Package

G Oracle Database Vault DVSYS.DBMS_MACUTL Package

H PL/SQL Interfaces to Oracle Database Vault

I Oracle Database Vault Security Guidelines

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 presents 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

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

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*

To download free release notes, installation documentation, updated versions of this guide, white papers, or other collateral, visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership/`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation/`

For OTN information specific to Oracle Database Vault, visit

`http://www.oracle.com/technology/deploy/security/db_security/database-vault/`

For frequently asked questions about Oracle Database Vault, visit

`http://www.oracle.com/database/docs/oracle-database-vault-faq.pdf`

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.

1

Introducing Oracle Database Vault

This chapter introduces you to Oracle Database Vault. It includes the following sections:

- What Is Oracle Database Vault?
- Components of Oracle Database Vault
- How Oracle Database Vault Addresses Compliance Regulations
- How Oracle Database Vault Addresses Insider Threats
- How Oracle Database Vault Allows for Flexible Security Policies
- How Oracle Database Vault Addresses Database Consolidation Concerns

What Is Oracle Database Vault?

Oracle Database Vault helps you address the most difficult security problems remaining today: protecting against insider threats, meeting regulatory compliance requirements, and enforcing separation of duty.

It provides a number of flexible features that can be used to apply fine-grained access control to your sensitive data. It hardens your Oracle Database instance and enforces industry standard best practices in terms of separating duties from traditionally powerful users. Most importantly, it protects your data from super privileged users but still allows them to maintain your Oracle databases. Oracle Database Vault can become an integral component of your enterprise.

You configure Oracle Database Vault to manage the security of an individual Oracle Database instance. You can install Oracle Database Vault on standalone Oracle Database installations, multiple Oracle homes, and in Oracle Real Application Clusters (RAC) environments.

For frequently asked questions about Oracle Database Vault, visit

`http://www.oracle.com/database/docs/oracle-database-vault-faq.pdf`

For Oracle Technology Network (OTN) information specific to Oracle Database Vault, visit

`http://www.oracle.com/technology/deploy/security/db_security/database-vault/`

Components of Oracle Database Vault

Oracle Database Vault has the following components:

- Oracle Database Vault Access Control Components
- Oracle Database Vault Administrator (DVA)
- Oracle Database Vault Configuration Assistant (DVCA)
- Oracle Database Vault DVSYS and DVF Schemas
- Oracle Database Vault PL/SQL Interfaces and Packages
- Oracle Database Vault and Oracle Label Security PL/SQL APIs
- Oracle Database Vault Reporting and Monitoring Tools

Oracle Database Vault Access Control Components

Oracle Database Vault enables you to create the following components to manage security for your database instance:

- **Realms:** A realm is a functional grouping of database schemas, objects, and roles that must be secured. For example, you can group a set of schemas, objects, and roles that are related to accounting, sales, or human resources. After you have grouped these into a realm, you can use the realm to control the use of system 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 4, "Configuring Realms" discusses realms in detail.
- **Command rules:** A command rule is a special rule 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 or not the statement is allowed. Chapter 6, "Configuring Command Rules" discusses command rules in detail.
- **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 secure. 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. 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 7, "Configuring Factors" discusses factors in detail.
- **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 5, "Configuring Rule Sets" discusses rule sets in detail.
- **Secure application roles:** A secure application role is a special Oracle role that can be enabled based on the evaluation of an Oracle Database Vault rule set. Chapter 8, "Configuring Secure Application Roles for Oracle Database Vault" discusses secure application roles in detail.

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 1-3 provides an overview.

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. Chapter 11, "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 Database Vault Administrator (DVA)

Oracle Database Vault Administrator is a Java application that is built on top of the Oracle Database Vault PL/SQL application programming interfaces (API). This application allows security managers who may not be proficient in PL/SQL to configure the access control policy through a user-friendly interface. 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 4 through Chapter 9 explain how to use Oracle Database Vault Administrator 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 11, "Oracle Database Vault Reports" explains Oracle Database Vault reporting.

Oracle Database Vault Configuration Assistant (DVCA)

To perform maintenance tasks on your Oracle Database Vault installation, use the command-line utility Oracle Database Vault Configuration Assistant (DVCA). For more information, see Appendix C, "Post-Installation Oracle Database Vault Procedures".

Oracle Database Vault DVSYS and DVF Schemas

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.

Appendix D, "Oracle Database Vault Objects" 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 allow 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 Appendix H, "PL/SQL Interfaces to Oracle Database Vault" and Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package" for more information.

Oracle Database Vault and Oracle Label Security PL/SQL APIs

Oracle Database Vault provides access control capabilities that can be integrated with Oracle Label Security. The Oracle Label Security database option is integrated with Oracle Enterprise Manager Database Control, which enables the security manager to define label security policy and apply it to database objects. Oracle Label Security also provides a collection of PL/SQL APIs that can be used by a database application developer to provide label security policy and protections.

See "Integrating Oracle Database Vault with Oracle Label Security" on page 9-2 for more information on how Oracle Database Vault works with Oracle Label Security. See also *Oracle Label Security Administrator's Guide* for more information about Oracle Policy Manager.

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 11, "Oracle Database Vault Reports" for more information about the reports that you can generate. Chapter 10, "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, 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, separation of duty features, command rules, and factors 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

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

Regulation	Potential Security Threat
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 Addresses Insider Threats

For many years, worms, viruses, and the external intruder (hacker) have been perceived as the biggest threats to computer systems. Unfortunately, what is often overlooked is the potential for trusted users or privileged users to steal or modify data.

Oracle Database Vault protects against insider threats 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 pass these conditions. This can prevent unauthorized access to the application data as well as 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 dozens of commands within 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 rules 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

In this way, you can carefully control and protect your system. You can disable and reenable command rules when you need to, and easily maintain them from one central location using Oracle Database Vault Administrator.

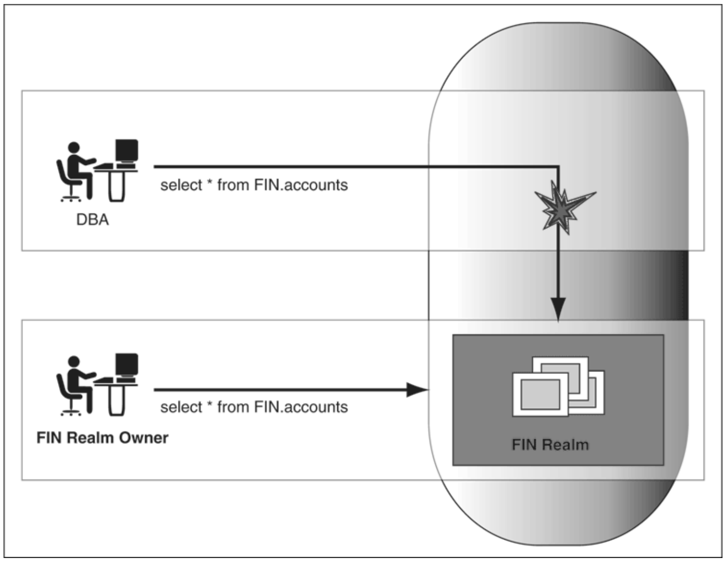
How Oracle Database Vault Addresses Database Consolidation Concerns

Oracle customers today still have hundreds and even thousands of databases distributed throughout the enterprise and around the world. However, database consolidation will continue as a cost-saving strategy in the coming years. 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 1–1 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 DBA from accessing the schemas that are protected by the FIN Realm. Although the DBA is the most powerful and trusted user, the DBA does not need access to application data residing within the database.
- **Separation of duties for application data access**: In this case, the FIN Realm Owner, created in Oracle Database Vault, has access to the FIN Realm schemas.

Figure 1–1 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 DBA, 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 application data. For example, a DBA who has the `SELECT ANY TABLE` privilege can be prevented from using that privilege to view application data.

2

What to Expect After You Install Oracle Database Vault

This chapter describes how your Oracle Database installation will change after you have installed Oracle Database Vault.

This chapter includes the following topics:

- Initialization and Password Parameter Settings That Change
- How Oracle Database Vault Restricts User Authorizations
- Using New Database Roles to Enforce Separation of Duties

See also Appendix I, "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 increase the security of your database configuration. If these changes adversely affect your organizational processes or database maintenance procedures, you can revert to the original settings.

Initialization Parameter Settings

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 operations issued by user SYS, and users connecting with SYSDBA or SYSOPER privileges. For more information about AUDIT_SYS_OPERATIONS, see *Oracle Database SQL Language Reference*.
OS_AUTHENT_PREFIX	ops$	Null string	The null string value disables SYSDBA operating system authentication. SYSOPER is still available for use. For more information about OS_AUTHENT_PREFIX, see *Oracle Database SQL Language 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*.
REMOTE_LOGIN_PASSWORDFILE	EXCLUSIVE	EXCLUSIVE	Oracle Database Vault uses password files to authenticate users. 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 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
REMOTE_OS_AUTHENT	FALSE	FALSE	Prevents remote clients from being authenticated with the value of the OS_AUTHENT_PREFIX parameter. This prevents a remote user from impersonating another operating system user over a network connection. For more information about REMOTE_OS_AUTHENT, see *Oracle Database Security Guide*.
REMOTE_OS_ROLES	FALSE	FALSE	Disables users who are connecting to the database through Oracle Net to have their roles authenticated by the operating system. This includes connections through a shared server configuration, as this connection requires Oracle Net. This restriction is the default because a remote user could impersonate another operating system user over a network connection. For more information about REMOTE_OS_ROLES, see *Oracle Database Security Guide*.
SQL92_SECURITY	FALSE	TRUE	Ensures that users have been granted the SELECT object privilege to execute such UPDATE or DELETE statements. 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 several 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.

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 duties*. 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 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 "Oracle Database Vault Roles" on page D-2 for detailed information about the roles created during the Oracle Database Vault installation. See also "Oracle Database Vault Accounts" on page D-7 for default accounts that are created and for suggestions of additional accounts that you may want to create.

3 Getting Started with Oracle Database Vault

This chapter provides a quick introduction to using Oracle Database Vault. You will learn how to start Oracle Database Vault Administrator, and then explore the basics of using Oracle Database Vault by creating a simple security configuration.

This chapter includes the following sections:

- Starting Oracle Database Vault Administrator
- Quick Start Tutorial: Securing a Schema from DBA Access

Starting Oracle Database Vault Administrator

This section describes how to start Oracle Database Vault Administrator.

> **Tip:** If you are using Oracle Enterprise Manager Database Control, you can configure it for an SSL (HTTPS) secure connection. For more information, see *Oracle Enterprise Manager Advanced Configuration* in Oracle Enterprise Manager 10*g* Release 10.1. To access this manual, visit Oracle Technology Network (OTN) at
>
> `http://www.oracle.com/technology/index.html`
>
> Follow these steps:
>
> 1. From the **Documentation** menu, select **Enterprise Manager**.
> 2. On the Oracle Enterprise Manager 10*g* Release Documentation page, under Other Releases, select the **Oracle Enterprise Manager Release 1 (10.1) Documentation** link.
> 3. On the Oracle Enterprise Manager 10*g* Release 1 (10.1) Documentation page, select either **PDF** or **HTML** for *Oracle Enterprise Manager Advanced Configuration*, whose part number is B12013-03.
> 4. In Chapter 4, "Enterprise Manager Security," go to "Configuring Security for the Database Control."

To start Oracle Database Vault Administrator:

1. From a browser, enter the following URL:

   ```
   https://host_name:port/dva
   ```

   In this specification:

   - *`host_name`*: The server where you installed Oracle Database Vault
   - *`port`*: The Oracle Enterprise Manager Console HTTP port number

   For example:

```
https://myserver:1158/dva
```

If you are unsure of the port number, open the *ORACLE_HOME*/*host_sid*/sysman/config/emd.properties file and search for REPOSITORY_URL.

If you cannot start Oracle Database Vault Administrator, check that the Oracle database console process is running.

- **On UNIX systems**: Navigate to the $ORACLE_HOME/bin directory and run the following command:

  ```
  ./emctl status dbconsole
  ```

  If you must start the dbconsole process, then run the following command:

  ```
  ./emctl start dbconsole
  ```

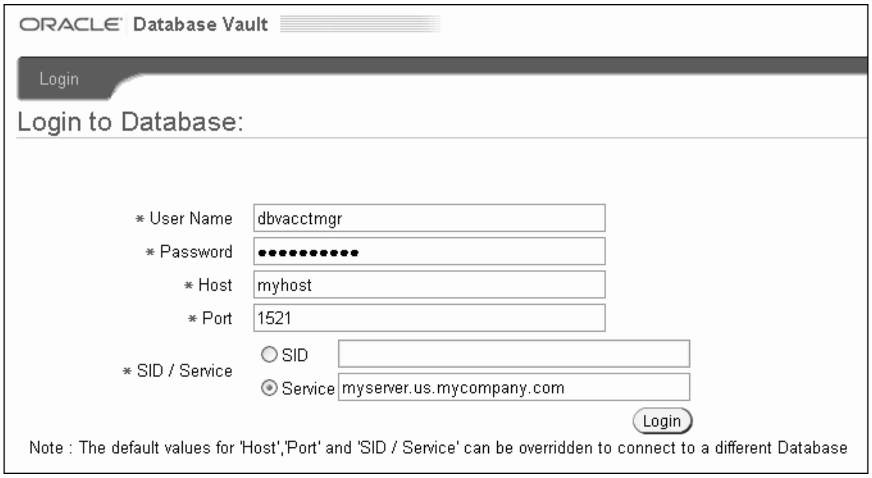
- **On Windows systems**: In the Administrative Services, select the Services utility, and then right-click the **OracleDBConsole*sid*** service. If necessary, select **Start** from the menu to start the database console.

Log files are in the following directory:

```
$ORACLE_HOME/sysman/log
```

2. Log in by using the Oracle Database Vault Owner account that you created during installation.

   By default, you cannot log in to Oracle Database Vault Administrator by using the SYS, SYSTEM, or other administrative accounts. You can log in if you have the DV_ADMIN or DV_OWNER roles.



Login to Database:

* User Name dbvacctmgr

* Password ●●●●●●●●●●●

* Host myhost

* Port 1521

* SID / Service ○ SID

● Service myserver.us.mycompany.com

Note : The default values for 'Host','Port' and 'SID / Service' can be overridden to connect to a different Database

By default, the login page enables you to log in to the default Oracle Database Vault installation. If you want to log in to a different Database Vault installation, enter the following values:

- **Host**: Enter the host name of the computer of the Oracle Database Vault installation you want.
- **Port**: Enter the port number.
- **SID/Service**: To connect using the server identification, select **SID** and then enter the server ID of the database you want, for example, orcl. To connect using the service information, select **Service**, and then enter the service identification in the following format:

server.domain

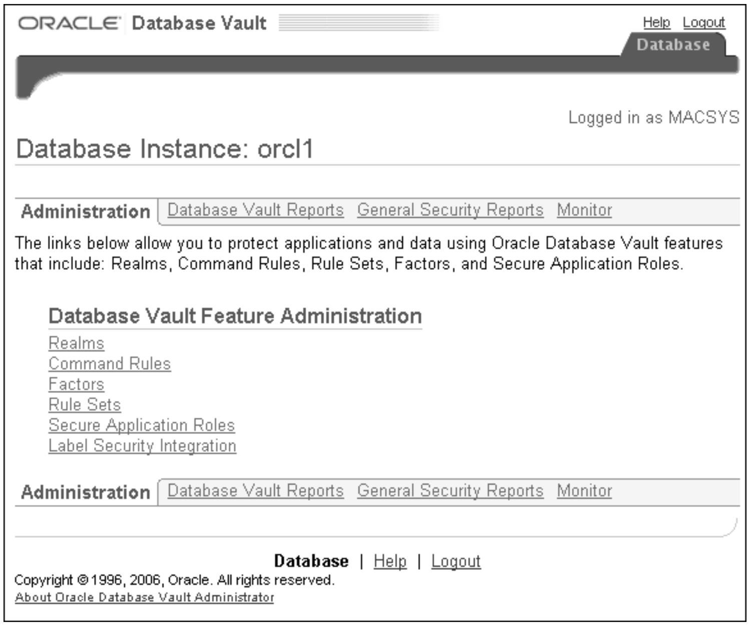
For example:

```
myserver.us.mycompany.com
```

To find the SID and service connection information, check the `tnsnames.ora` file. By default, this file is located in *`ORACLE_BASE`*`/`*`ORACLE_HOME`*`/network/admin`.

Figure 3–1 shows the Oracle Database Vault Administrator home page, which appears after you log in.

Figure 3–1 Oracle Database Vault Administrator Home Page



Database Instance: orcl1

The links below allow you to protect applications and data using Oracle Database Vault features that include: Realms, Command Rules, Rule Sets, Factors, and Secure Application Roles.

Database Vault Feature Administration

Copyright © 1996, 2006, Oracle. All rights reserved.

Quick Start Tutorial: Securing a Schema from DBA Access

In this tutorial, you will 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 will 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 will 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.

You will follow these steps:

- Step 1: Adding the SYSTEM User to the Data Dictionary Realm
- Step 2: Log On as SYSTEM to Access the HR Schema
- Step 3: Create a Realm
- Step 4: Secure the EMPLOYEES Table in the HR Schema

- Step 5: Create an Authorization for the Realm
- Step 6: Test the Realm
- Step 7: Run a Report
- Step 8: Remove the Components for This Example

Step 1: Adding the SYSTEM User to the Data Dictionary Realm

In this tutorial, the `SYSTEM` user will grant `ANY` privileges to a new user account, `SEBASTIAN`. In order to do this, `SYSTEM` will need to be included in the Oracle Data Dictionary realm.

To include `SYSTEM` in the Oracle Data Dictionary realm:

1. Log in to Oracle Database Vault Administrator using a database account that has been granted the `DV_OWNER` role.

   "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

2. In the Administration page of Oracle Database Vault Administrator, under Database Vault Feature Administration, click **Realms**.
3. In the Realms page, select **Oracle Data Dictionary** from the list and then click **Edit**.
4. In the Edit Realm: Oracle Data Dictionary page, under Realm Authorizations, click **Create**.
5. In the Create Realm Authorization Page, from the Grantee list, select **SYSTEM [USER]**.
6. For Authorization Type, select **Owner**.
7. Leave Authorization Rule Set at **<Non Selected>**.
8. Click **OK**.

   In the Edit Realm: Oracle Data Dictionary page, `SYSTEM` should be listed as an owner under the Realm Authorizations.

9. Click **OK** to return to the Realms page.
10. To return to the Administration page, click the **Database Instance *instance_name*** link over Realms.

Step 2: Log On as SYSTEM to Access the HR Schema

Log in to SQL*Plus with administrative privileges and access the `HR` schema. You will log on using the `SYSTEM` account.

```
$ sqlplus system
Enter password: password
SQL> SELECT FIRST_NAME, LAST_NAME, SALARY FROM HR.EMPLOYEES WHERE ROWNUM < 10;

FIRST_NAME           LAST_NAME                     SALARY
-------------------- ------------------------- ----------
Donald               OConnell                        2600
Douglas              Grant                           2600
Jennifer             Whalen                          4400
Michael              Hartstein                      13000
Pat                  Fay                             6000
Susan                Mavris                          6500
Hermann              Baer                           10000
Shelley              Higgins                        12000
```

```
William            Gietz                           8300

9 rows selected.
```

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.

Step 3: 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.

Follow these steps:

1. If you have not done so already, return to the Administration page by clicking the **Database Instance *instance_name*** link over Realms.
2. In the Administration page, under Database Vault Feature Administration, click **Realms**.
3. In the Realms page, click **Create**.
4. In the Create Realm page, under General, enter HR Realm after **Name**.
5. After Status, ensure that **Enabled** is selected so that the realm can be used.
6. Under Audit Options, ensure that **Audit On Failure** is selected so that you can create an audit trial later on.
7. Click **OK**.

   The Realms Summary page appears, with HR Realm in the list of realms.

Step 4: Secure the EMPLOYEES Table in the HR Schema

At this stage, you are ready to add the EMPLOYEES table in the HR schema to the secured objects of the HR realm.

Follow these steps:

1. In the Realms page, select **HR Realm** from the list and then click **Edit**.
2. In the Edit Realm: HR Realm page, scroll to Realm Secured Objects and then click **Create**.
3. In the Create Realm Secured Object page, enter the following settings:
   - **Object Owner:** Select HR from the list.
   - **Object Type:** Select %.
   - **Object Name:** Enter EMPLOYEES.
4. Click **OK**.
5. In the Edit Realm: HR Realm page, click **OK**.

Step 5: Create an Authorization for the Realm

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. After you authorize him for the realm, SEBASTIAN will be able to view and modify the EMPLOYEES table.

First, in SQL*Plus, create a user account that has the SELECT privilege on the EMPLOYEES table in the HR schema. To do so, log in SQL*Plus using an account that has the DV_ACCTMGR role, which is used for account management. When you installed Oracle Database Vault, the account that you created during the installation process was granted the DV_ACCTMGR role. For example, suppose you had created an account called DBVACCTMGR during the installation process. You would log in using the DBVACCTMGR account, and then create the user account as follows:

```
SQL> CONNECT dbvacctmgr
Enter password: password
SQL> CREATE USER SEBASTIAN IDENTIFIED BY seb_456987;
User created.
SQL> GRANT CONNECT TO SEBASTIAN;
Grant succeeded.
SQL> CONNECT SYSTEM
Enter password: password
Connected.
SQL> GRANT SELECT ANY TABLE TO SEBASTIAN;
Grant succeeded.
```

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

At this point SEBASTIAN still can query the EMPLOYEES table, because the GRANT statement overrides realm protection. If the GRANT statement were generic, for example, SELECT ANY TABLE, then Oracle Database Vault restricts access to all tables protected by realm, including HR.EMPLOYEES.

Next, authorize user SEBASTIAN to have access to the HR Realm as follows:

1. In the Realms page, select the **HR Realm** in the list of realms, and then click **Edit**.
2. In the Edit Realm: HR Realm page, scroll down to Realm Authorizations and then click **Create**.
3. In the Create Realm Authorization page, under Grantee, select **SEBASTIAN** from the list.

   If SEBASTIAN does not appear in the list, select the **Refresh** button in your browser.

   SEBASTIAN will be the only user who will have access to the EMPLOYEES table in the HR schema.
4. Under Authorization Type, select **Owner**.

   The Owner authorization allows the user SEBASTIAN in the HR realm to manage the database roles protected by HR, as well as create, access, and manipulate objects within the realm. In this case, the HR user and SEBASTIAN will be the only persons allowed to view the EMPLOYEES table.
5. Under Authorization Rule Set, select **<Not Assigned>**, because rule sets are not needed to govern this realm.
6. Click **OK**.

Step 6: 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.

Log in to SQL*Plus as SYSTEM, and then try accessing the salary information in the EMPLOYEES table again:

```
sqlplus system
Enter password: password
SQL> SELECT FIRST_NAME, LAST_NAME, SALARY FROM HR.EMPLOYEES WHERE ROWNUM <10;

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

As you can see, SYSTEM no longer has access to the salary information in the EMPLOYEES table.

However, user SEBASTIAN does have access to the salary information in the EMPLOYEES table:

```
SQL> CONNECT SEBASTIAN
Enter password: seb_456987
Connected.
SQL> SELECT FIRST_NAME, LAST_NAME, SALARY FROM HR.EMPLOYEES WHERE ROWNUM <10;

FIRST_NAME           LAST_NAME                     SALARY
-------------------- ------------------------- ----------
Donald               OConnell                        2600
Douglas              Grant                           2600
Jennifer             Whalen                          4400
Michael              Hartstein                      13000
Pat                  Fay                             6000
Susan                Mavris                          6500
Hermann              Baer                           10000
Shelley              Higgins                        12000
William              Gietz                           8300

9 rows selected.
```

Step 7: Run a Report

Because you enabled auditing on failure for the HR Realm, you can generate a report to find any security violations such as the one you attempted in Step 6: Test the Realm.

Follow these steps:

1. In the Oracle Database Vault Administrator home page, click **Database Vault Reports**.

   To run the report, log in using an account that has the DV_OWNER, DV_ADMIN, or DV_SECANALYST role. Note that user SEBASTIAN cannot run the report, even if it affects his own realm. "Oracle Database Vault Roles" on page D-2 describes these roles in detail.

2. In the Database Vault Reports page, scroll down to Database Vault Auditing Reports and select **Realm Audit**.

3. Click **Run Report**.

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

Step 8: Remove the Components for This Example

Follow these steps:

1. Remove the SYSTEM account from the Data Dictionary Realm.
   1. Ensure that you are logged on to Oracle Database Vault Administrator using a database account that has been granted the DV_OWNER role.
   2. From the Administration page, select **Realms**.
   3. From the list of realms, select Oracle Data Dictionary, and then click **Edit**.
   4. Under Realm Authorizations, select SYSTEM.
   5. Click **Remove**, and in the Confirmation window, click **Yes**.
2. Delete the HR Realm.
   1. In the Realms page, select HR Realm from the list of realms.
   2. Click **Remove**, and in the Confirmation window, click **Yes**.
3. Drop user SEBASTIAN.

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

   ```
   sqlplus dbvacctmgr
   Enter password: password
   SQL> DROP USER SEBASTIAN;
   User dropped.
   ```

4

Configuring Realms

This chapter describes how to create and maintain realms. It includes the following sections:

What Are Realms?

A *realm* is a functional grouping of database schemas and 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 classifying schemas and roles into functional groups, you can control the ability to use system privileges against these groups and prevent unauthorized data access by the DBA 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.

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, after you install Oracle Database Vault, you can create a realm to protect all existing database schemas that are used in an accounting department. The realm will prohibit 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" on page 4-12 for more information.

This chapter explains how to configure realms by using Oracle Database Vault Administrator. To configure realms by using the PL/SQL interfaces and packages provided by Oracle Database Vault, refer to the following appendixes:

- Appendix H, "PL/SQL Interfaces to Oracle Database Vault"
- Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package"

Default Realms

Oracle Database Vault provides the following default realms:

- **Database Vault Account Management**: Defines the realm for the administrators who manage and create database accounts and database profiles.
- **Oracle Data Dictionary**: Defines the realm for the following Oracle Catalog schemas.

  `ANONYMOUS`
  `BI`
  `CTXSYS`
  `DBSNMP`
  `EXFSYS`
  `MDDATA`
  `MDSYS`
  `MGMT_VIEW`
  OUTLN
  SYS
  SYSMAN
  SYSTEM

  This realm also controls the ability to grant system privileges and database administrator roles.
- **Oracle Database Vault**: Defines the realm for the Oracle Database Vault schemas (`DVSYS`, `DVF`, and `LBACSYS`), such as configuration and roles information.
- **Oracle Enterprise Manager**: Defines the realm for Oracle Enterprise Manager accounts (`SYSMAN` and `DBSNMP`) to access database information

Creating a Realm

In general, to enable realm protection, you first create the realm itself, and then you edit the realm to include realm secured objects, roles, and authorizations. "Guidelines for Designing Realms" on page 4-10 provides advice on creating realms.

To create a realm:

1. Log in to Oracle Database Vault Administrator using a database account that has been granted the `DV_OWNER` role.

   At a minimum, you must have the `DV_ADMIN` role. "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.
2. In the Administration page, under Database Vault Feature Administration, click **Realms**.
3. In the Realms page, click **Create**.
4. In the Create Realm page, enter the following settings:
   - Under General:

  - **Name**: Enter a name for the realm. It can contain up to 90 characters in mixed-case. This attribute is mandatory.
  - **Description**: Enter a brief description of the realm. The description can contain up to 1024 characters in mixed-case. This attribute is optional.
  - **Status**: Select either **Enabled** or **Disabled** to enable or disable the realm during run time. A realm is enabled by default. This attribute is mandatory.
- Under Audit Options, select one of the following:
  - **Audit Disabled**: Does not create an audit record.
  - **Audit On Failure** (default): 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.

5. Click **OK**.

   The Realms Summary page appears, listing the new realm that you created.

After you create a new realm, you are ready to add schema and database objects to the realm for realm protection, and to authorize users and roles to access the realm. To do so, you edit the new realm and then add its objects and its authorized users.

See Also:

- "Editing a Realm" on page 4-3
- "Creating Realm-Secured Objects" on page 4-4
- "Defining Realm Authorization" on page 4-5

Editing a Realm

To edit a realm:

1. In the Oracle Database Vault Administration page, select **Realms**.
2. In the Realm page, select the realm that you want to edit.
3. Click **Edit**.
4. Modify the realm as necessary, and then click **OK**.

See Also:

- "Creating a Realm" on page 4-2 to modify the settings created for a new realm
- "Creating Realm-Secured Objects" on page 4-4 to add or modify realm secured objects
- "Defining Realm Authorization" on page 4-5 to add or modify the realm authorizations

Creating 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 permissions. 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, the commands 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.

You can manage the objects secured by a realm from the Edit Realm page, which lets you create, edit, and delete realm secured objects.

To create a realm secured object:

1. In the Oracle Database Vault Administration page, select **Realms**.
2. In the Realms page, select the realm you want, and then select **Edit**.
3. In the Edit Realm page, under Realm Secured Objects, do one of the following:
   - To create a new realm-secured object, select **Create**.
   - To modify an existing object, select it from the list and then select **Edit**.
4. In the Create Realm Secured Object page, enter the following settings:
   - **Object Owner**: From the list, select the name of the database schema owner. This attribute is mandatory.
   - **Object Type**: From the list, select the object type of the database object, such as TABLE, INDEX, or ROLE. This attribute is mandatory.

     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 will affect, or enter % to specify all objects (except roles) for the object owner that you have specified. However, you cannot use wildcard characters with text such to specify multiple object names, for example, EMP_% to specify all tables beginning with the characters EMP_. Nor can you use the wildcard character to select multiple roles; you must enter role names individually. This attribute is mandatory.

     By default, the **Object Name** field contains the % wildcard character to include all objects within the specified **Object Type** and **Object Owner**. Note that the % wildcard character applies to objects that do not yet exist as well as currently existing objects. Note also that the % wildcard character does not apply to roles. If you want to include more than one role, you must specify each role separately.
5. Click **OK**.

For example, to secure the EMPLOYEES table in the HR schema, you would enter the following settings in the Create Realm Secured Object page:

- **Object Owner**: `HR`
- **Object Type**: `TABLE`
- **Object Name**: `EMPLOYEES`

Editing a Realm-Secured Object

To edit a realm-secured object:

1. Select the object under Realm Secured Objects in the Edit Realm page.
2. Click **Edit**.
3. In the Edit Realm Secured Object page, edit the attributes as required.
4. Click **OK**.

Deleting a Realm-Secured Object

To delete a realm-secured object:

1. Select the object under Realm Secured Objects in the Edit Realm page.
2. Click **Remove**.

   A confirmation page is displayed.
3. Click **Yes**.

   This dissociates the object from the realm and unsecures it. (The regular database protections still apply.) However, it does not remove the object from the database.

Defining Realm Authorization

Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms. A realm authorization can be an account or role that is authorized to use its system privileges in the following situations:

- When the user needs to create or access realm-secured objects
- When a user needs to 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:

- The authorization that you set up here does not affect regular users who have normal direct object privileges to the database objects that are protected by realms.
- 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.
- A realm owner, but not a realm participant, can grant or revoke realm secured database roles 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 options of a realm authorization.

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 11-4.

To create a realm authorization:

1. In the Oracle Database Vault Administration page, select **Realms**.
2. In the Realms page, select the realm you want, and then select **Edit**.
3. In the Edit Realm page, under Realm Authorizations, do one of the following:
   - To create a new realm authorization, select **Create**.
   - To modify an existing realm authorization, select it from the list and then select **Edit**.
4. Click **Create** under Realm Authorizations in the Edit Realm page.
5. In the Create Realm Authorization page, enter the following settings:
   - **Grantee**: From the list, select the Oracle 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. (Note that you cannot select yourself or any account that has been granted the `DV_ADMIN` or `DV_OWNER` roles from this list to grant yourself realm ownership.)
   - **Authorization Typ**e: Select either of the following. This attribute is mandatory.
     - **Participant** (default): 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. A realm can have more than one participant.
     - **Owner**: This account or role has the same privileges as the realm participant, plus the authorization to grant or revoke realm-secured database roles. A realm can have more than one owner.
   - **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 5-5 for more information about defining rules to govern the realm authorization.

     Any auditing and custom event handling associated with the rule set will occur as part of the realm authorization processing.
6. Click **OK**.

Editing a Realm Authorization

To edit a realm authorization:

1. Select the realm authorization under Realm Authorizations in the Edit Realm page.
2. Click **Edit**.

   The Edit Realm Authorization page is displayed.
3. Edit the attributes as required.
4. Click **OK**.

Deleting a Realm Authorization

To delete a realm authorization:

1. Select the realm authorization under Realm Authorizations in the Edit Realm page.
2. Click **Remove**.

   A confirmation page is displayed.
3. Click **Yes**.

Disabling and Enabling a Realm

By default, when you create a realm, it is enabled. You can disable a realm, for example, for system maintenance such as patch updates, and then enable it again afterward.

To disable or enable a realm:

1. In the Oracle Database Vault Administration page, select **Realms**.
2. In the Realms page, select the realm you want to disable or enable, and then select **Edit**.
3. In the Edit Realm page, under Status in the General section, select either **Disabled** or **Enabled**.
4. Click **OK**.

Deleting a Realm

Before you delete a realm, you can locate the various references to it by querying the realm-related Oracle Database Vault views. See "Oracle Database Vault Public Views" on page D-10 for more information.

To delete a realm:

1. In the Oracle Database Vault Administration page, select **Realms**.
2. In the Realms page, select the realm you want to delete, and then select **Remove**.
3. In the Confirmation page, click **Yes**.

   Oracle Database Vault deletes the configuration for a realm (header, secure objects, and authorizations). It does not delete the rule sets within the realm.

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 customer-defined realm, the following actions occur:

1. Is the database account using a system privilege to execute the SQL statement?

   If yes, then go to Step 2. If no, then go to Step 6. If the session has object privileges on the object in question for SELECT, EXECUTE, and DML only, then the realm protection is not enforced. Realms protect against the use of the any system privileges 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.

2. Does the SQL statement affect objects secured by a realm?

   If yes, then go to Step 3. If no, then realms do not affect the SQL statement; go to Step 6. If the object affected by the command is not secured in any realms, then realms do not affect the SQL statement being attempted.

3. Is the database account a realm owner or realm participant?

   If yes, and 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, the session must be authorized as the realm owner directly or indirectly through a protected role in the realm. Then go to Step 4. Otherwise, realm violation occurs and the statement is not allowed to succeed. Note that SYS is the only realm owner in the default Oracle Data Dictionary Realm, and only SYS can grant system privileges to a database account or role.

4. Is the realm authorization for the database account conditionally based on a rule set? If yes, then go to Step 5. If no, then go to Step 6.

5. Does the rule set evaluate to true?

   If yes, then go to Step 6. If no, then there is a realm violation, so the SQL statement is not allowed to succeed.

6. 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, 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 as well as direct 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 can use any privilege on objects protected by the realm. 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:

- Realms do not protect views that had been created on a table before the table was added to a realm. To protect the view, explicitly add it to the realm.
- For invoker's right procedures that access realm protected objects, the invoker of the procedure must be authorized to the realm.
- The execution of PL/SQL procedures that are owned by SYS are subject to the Oracle Data Dictionary realm enforcement. (The Oracle Data Dictionary realm is one of the default realms provided by Oracle Database Vault. See "Default Realms" on page 4-2 for more information.) However, the session must have EXECUTE privilege on the procedure as normally required in the Oracle database.
- Java stored procedures are not protected by a realm, but the data objects that a Java stored procedure accesses can be protected by the realm. You should create the Java stored procedure with invoker's rights so that someone who is really authorized can see the protected data.

How Authorizations Work in a Realm

Realms protect data from access through system privileges; realms do not give additional privileges to its owner or participants. The realm authorization provides a runtime mechanism to check logically if a user's command is allowed 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 direct 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.

Example 4–1 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 4–1 Unauthorized User Trying to Create a Table

```
SQL> CREATE TABLE HR.demo2 (col1 NUMBER(1));

ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-20401: Realm Violation on table HR.DEMO2
ORA-06512: at "DVSYS.AUTHORIZE_EVENT", line 35
ORA-06512: at line 13
```

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 4–2 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 4–2 Unauthorized User Trying to Use the DELETE ANY TABLE Privilege

```
SQL> DELETE FROM HR.employees WHERE empno = 8002;
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 4–3 shows how an authorized user can perform standard tasks allowed within the realm.

Example 4–3 Authorized User Performing DELETE Operation

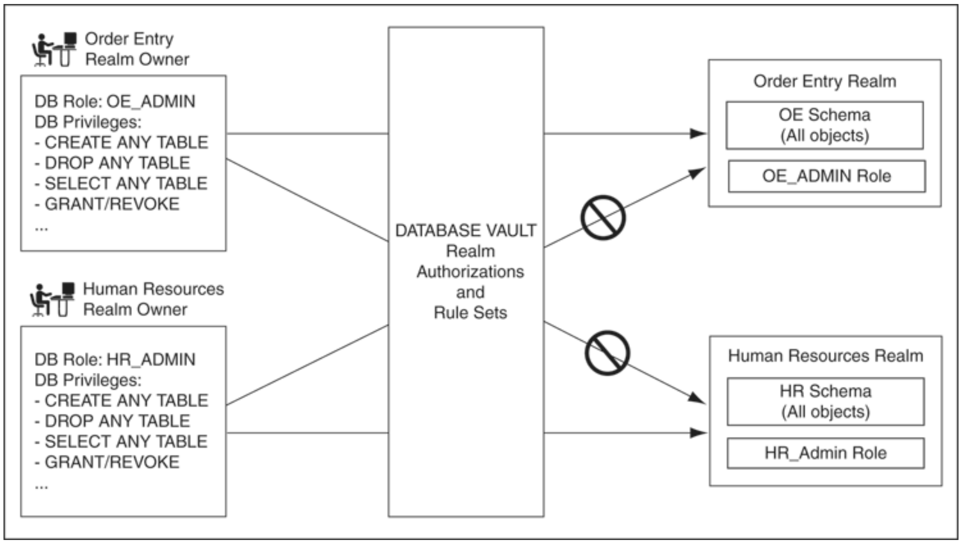
```
SQL> DELETE FROM HR.employees WHERE empno = 8002;

1 row deleted.
```

Example of How Realms Work

Figure 4–1 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 Data Dictionary realm. Any unauthorized attempt to use a system privilege to access realm-protected objects will create 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 4–1 How Authorizations Work for Realms and Realm Owners



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 4-7 explains the steps that Oracle Database Vault takes to process SQL statements that affect objects in a realm. "How Command Rules Work" on page 6-6 describes how command rules are processed.

Guidelines for Designing Realms

Follow these guidelines when 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 more than one realm and an account or role can be authorized in more than one realm.

  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.

- 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 need to 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 data dictionary realm so that this user can grant other users ANY privileges, if they need them.

- Sometimes you must perform imports and exports of data protected by a realm, for example, when using Oracle Data Pump. As the realm owner, perform the following steps. Be sure to audit the import and export activity using the techniques described in this chapter, whenever possible.

  1. Add the account that will perform the imports and exports to the realm and assign it as the realm participant of the Oracle Data Dictionary realm. Have it use this status during the time frame of the data transfer, with a rule set governing the authorization that will perform the auditing.
  2. Add the account that will perform the imports and exports to be the realm participant of the realm protecting the data for the time frame of the data transfer with a rule set governing the authorization that will perform auditing.
  3. On import, if the schema accounts that contain the realm tables do not exist in the target database, grant the importing user the DV_ACCTMGR role for the time frame that the import will occur. When the import is complete, you can revoke the grant of the role.

How Realms Affect Performance

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, but be careful not to then put a realm around that entire schema, thus having a realm around realms.

Auditing affects performance. To achieve the best performance, Oracle recommends that you use fine-grained auditing rather than auditing all operations.

You can check the system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Database Control, which is installed by default with Oracle Database), `Statspack`, and `TKPROF`. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Database Control, refer to its online Help. *Oracle Database Performance Tuning Guide* describes the `Statspack` and `TKPROF` utilities.

Related Reports

Table 4–1 lists Oracle Database Vault reports that are useful for analyzing realms. See Chapter 11, "Oracle Database Vault Reports" for information about how to run these reports.

Table 4–1 Reports Related to Realms

Report	Purpose
"Realm Audit Report" on page 11-4	To audit records generated by the realm protection and realm authorization operations
"Realm Authorization Configuration Issues Report" on page 11-4	To find authorization configuration information, such as incomplete or disabled rule sets, or nonexistent grantees or owners that may affect the realm
"Rule Set Configuration Issues Report" on page 11-4	To find rule sets that do not have rules defined or enabled, which may affect the realms that use them
"Object Privilege Reports" on page 11-6	To find object privileges that the realm affects
"Privilege Management - Summary Reports" on page 11-9	To find information about grantees and owners for a realm
"Sensitive Objects Reports" on page 11-7	To find objects that the command rule affects

5 Configuring Rule Sets

This chapter describes how to create and manage rule sets and the rules within them. It includes the following sections:

- What Are Rule Sets?
- Default Rule Sets
- Creating a Rule Set
- Configuring or Editing a Rule Set
- Creating a Rule to Add to a Rule Set
- Deleting a Rule Set
- How Rule Sets Work
- Example of How Rule Sets Work
- Guidelines for Designing Rule Sets
- How Rule Sets Affect Performance
- Related Reports

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" on page 5-13 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 the following appendixes:

- Appendix H, "PL/SQL Interfaces to Oracle Database Vault"
- Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package"

Default Rule Sets

By default, Oracle Database Vault provides the following selections for rule sets:

- **Allow Sessions**: Controls the ability to create a session in the database. This rule set enables you to add rules to control database logins using the CONNECT command rule. The CONNECT command rule is useful to control or limit `SYSDBA` access to programs that require its use. This rule set is not populated.
- **Allow System Parameters**: Controls the ability to set system initialization parameters. See *Oracle Database Reference* for detailed information about initialization parameters.
- **Can Grant VPD Administration**: Controls the ability to grant the `GRANT EXECUTE` or `REVOKE EXECUTE` privileges on the Oracle Virtual Private Database `DBMS_RLS` package, with the `GRANT` and `REVOKE` statements.
- **Can Maintain Accounts/Profiles**: Controls the roles that manage user accounts and profiles, through the `CREATE USER`, `DROP USER`, `CREATE PROFILE`, `ALTER PROFILE`, or `DROP PROFILE` statements.
- **Can Maintain Own Account**: Allows the accounts with the `DV_ACCTMGR` role to manage user accounts and profiles with the `ALTER USER` statement. Also allows individual accounts to change their own password using the `ALTER USER` statement.
- **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. You also can create the rule set without any rules to use as a template for future rule sets.

See also the following sections:

- "Guidelines for Designing Rule Sets" on page 5-12 for advice on designing rule sets
- "Oracle Database Vault PL/SQL Rule Functions" on page H-8 for a set of functions that you can use in rule expressions
- "Rule Set Configuration Issues Report" on page 11-4 to check the configuration of the rule sets for your database

To create a rule set:

1. Log in to Oracle Database Vault Administrator using a database account with the `DV_OWNER` role.

   At a minimum, you must have the `DV_ADMIN` role. "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

2. In the Administration page, under Database Vault Feature Administration, click **Rule Sets**.
3. In the Rule Sets page, click **Create**.
4. In the Create Rule Set page, enter the following settings, and then click **OK**:
   - General
   - Audit Options
   - Error Handling Options

General

Enter the following settings:

- **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.
- **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.
- **Status**: Select either **Enabled** or **Disabled** to enable or disable the rule set during run time. Rule sets are enabled by default. This attribute is mandatory.
- **Evaluation Options**: If you plan to assign more than one rule to a rule set, select one of the following settings:
  - **All True** (default): 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

Select from the following options to determine when an audit record is created for the rule set. This attribute is mandatory. The settings are:

- **Audit Disabled**: Does not create an audit record under any circumstances.
- **Audit On Failure** (default): 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.

The Oracle Database Vault audit trail contains the fields `Rule_Set_Name` and `Rule_Set_ID`. These fields are populated when a rule set is associated with a realm authorization and a command authorization, and the rule set is configured to audit under some circumstances.

See Appendix A, "Oracle Database Vault Auditing Policies" for more information. Table A–2, " Audit Trail Format" on page A-5 lists the information that is audited.

Error Handling Options

Enter the following settings to control the messaging to the database session when the rule set evaluates to false or one of the associated rules contains an invalid PL/SQL expression:

- **Fail Options**: Select either **Show Error Message** (the default) 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 negative number in the range of -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 the following error code:

  ```
  ORA-01031: Insufficient privileges
  ```

- **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** (default): 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 Appendix H, "PL/SQL Interfaces to Oracle Database Vault".

  Write the expression as a fully qualified procedure (such as `schema.procedure_name`). Do not include complete SQL statements. If you are using application package procedures or standalone procedures, you must provide DVSYS with the `GRANT 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.

When you define the expression in the user interface that uses one of these two formats, put the expression in the following form:

```
myschema.my_ruleset_handler
```

After you create a rule set, you are ready to create rules to attach to the rule set. To do so, you edit the new rule set, and then define its rules.

See Also:

- "Configuring or Editing a Rule Set" on page 5-5
- "Creating a Rule to Add to a Rule Set" on page 5-5

Configuring or Editing a Rule Set

To configure or edit a rule set:

1. In the Oracle Database Vault Administration page, select **Rule Sets**.
2. In the Rule Set page, select the rule set that you want to edit.
3. Click **Edit**.
4. Modify the rule set as necessary, and then click **OK**.

See Also:

- "Creating a Rule Set" on page 5-2 to modify the settings created for a new rule set
- Creating a Rule to Add to a Rule Set on page 5-5 to add or modify rule for the rule set

Creating a Rule to Add to a Rule Set

After you create a new rule set, you can associate it with one or more rules. When you create a new rule, it 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.

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 5-8 for information on how rules are evaluated, how to nest rules, and how to create rules that exclude a particular user, such as a super system administrator.

Creating a New Rule

To create and add a rule to a rule set:

1. In the Oracle Database Vault Administration page, select **Rule Sets**.
2. In the Rule Sets page, select the rule set to which you want to create and add a rule, and then select **Edit**.
3. In the Edit Rule Set Page, scroll down to Rules Associated To The Rule Set and select **Create**.
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.
- **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 255 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 complete SQL statements.

    If you want to use application package functions or standalone functions, you must grant the `DVSYS` account the `GRANT 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') != 'SQL*Plus'
    ```

    You would test this expression as follows:

    ```
    SELECT SYS_CONTEXT('USERENV','SESSION_USER') FROM DUAL;
    ```

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

  - "Oracle Database Vault PL/SQL Rule Functions" on page H-8
  - Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package"
  - Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package"

  For additional examples of expressions, see the rule defined in the rule sets provided with Oracle Database Vault. "Default Rule Sets" on page 5-2 lists these rule sets.

5. Click **OK**.

   The Edit Rule Set page appears. By default, the new rule is added to the rule set.

Editing a Rule

The changes you make to a rule apply to all rule sets that include the rule.

To edit a rule:

1. In the Edit Rule Set page, scroll to Rules Associated To The Rule Set.
2. Select the rule you want to edit and click **Edit**.
3. In the Edit Rule page, modify the rule as necessary.
4. Click **OK**.

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 "Oracle Database Vault Public Views" on page D-10 for more information.

To remove a rule from a rule set:

1. In the Edit Rule Set page, scroll to Rules Associated To The Rule Set.
2. Select the rule you want to delete and click **Remove**.
3. In the Confirmation page, click **Yes**.

After you remove the rule from the rule set, it still exists. If you want, you can associate it with other rule sets. If you want to delete the rule, use the `DVSYS.DBMS_MACADM.DELETE_RULE` function, described in "Rule Set Functions Within DVSYS.DBMS_MACADM" on page E-10.

Adding Existing Rules to a Rule Set

To add existing rules to a rule set:

1. In the Rule Sets page, select the rule set that you want to add rules to, and then select **Edit**.
2. Under Rules Associated To The Rule Set, select **Add Existing Rules**.
3. 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.
4. Click **OK**.

Deleting a Rule Set

Before you delete a rule set, you can locate the various references to it by querying the rules-related Oracle Database Vault views. See "Oracle Database Vault Public Views" on page D-10 for more information.

To delete a rule set:

1. If other Database Vault objects, such as command rules, reference the rule set, then remove the reference.

   You can delete a rule set only if no other Database Vault objects are referencing it.
2. In the Oracle Database Vault Administration page, select **Rule Sets**.
3. In the Rule Set page, select the rule set that you want to remove.
4. Click **Remove**.
5. In the Confirmation page, click **Yes**.

   The rule set is deleted. However, the rules associated with the rule set are not deleted.

How Rule Sets Work

This section describes how rule sets work in the following ways:

- How Oracle Database Vault Evaluates Rules
- Improving Performance by Setting the Order in Which Rules Appear in a Rule Set
- Nesting 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 fails the evaluation, 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.

Improving Performance by Setting the Order in Which Rules Appear in a Rule Set

Generally speaking, the order in which rules appear within a rule set does not affect the final outcome: the rule set either permits or prevents an action. However, the order can affect performance. You can place multiple rules within a single rule and prioritize them by using the `AND` or `OR` operator to improve the performance of the rule.

Nesting 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 PL/SQL Factor Functions" on page H-5 for more information. Chapter 7 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, the super system administrator. The rule definition for this type of rule can be as follows:

```
SYS_CONTEXT('USERENV','SESSION_USER') = 'SUPERADMIN_USER' OR additional_rule
```

If the current user is the super system administrator, then the system evaluates the rule to true without evaluating *`additional_rule`*. If the current user is not the super system administrator, then the evaluation of the rule depends on the evaluation of *`additional_rule`*.

Example of How Rule Sets Work

In the following example, you will create a variation of the Can Maintain Tables rule set, which was created to control when table objects can be created, altered, or dropped. In your rule set, you will a PL/SQL procedure that sends an e-mail security alert if the rule is violated.

You will follow these steps to complete this example:

- Step 1: Install and Configure the UTL_MAIL PL/SQL Package
- Step 2: Create an E-mail Security Alert PL/SQL Procedure
- Step 3: Create an Oracle Database Vault Rule Set That Uses the E-mail Security Alert
- Step 4: Test the E-mail Security Alert

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

Follow these steps:

1. Log on to SQL*Plus and install the `UTL_MAIL` package.

   ```
   sqlplus "sys / as sysdba"
   Enter password: password
   SQL> @$ORACLE_HOME/rdbms/admin/utlmail.sql
   SQL> @$ORACLE_HOME/rdbms/admin/prvtmail.plb
   ```

   The `UTL_MAIL` package enables you to manage e-mail. See *Oracle Database PL/SQL Packages and Types Reference* for more information on `UTL_MAIL`.

2. Add the following configuration settings to the initialization configuration file, `init`*`sid`*`.ora`:

   ```
   SMTP_SERVER="my_server.my_company.com"
   SMTP_DOMAIN="my_company.com"
   SMTP_SERVER_OUT="stmail.my_company.com"
   ```

   Replace *`my_server`* and *`my_company.com`* with the name of your server and company. Enclose these settings in quotation marks.

Step 2: Create an E-mail Security Alert PL/SQL Procedure

Follow these steps:

1. Connect to SQL*Plus as the Oracle Database Owner (`DV_OWNER`) account. For example:

   ```
   CONNECT dbvacctmgr
   Enter password: password
   ```

2. Create the following procedure:

   ```
   CREATE OR REPLACE PROCEDURE email_alert AS
   msg varchar2(20000) := 'Realm violation occurred for the JSMITH rule set. The
   time is: ';

   BEGIN
     msg := msg||to_char(SYSDATE, 'Day DD MON, YYYY HH24:MI:SS');

   UTL_MAIL.SEND (
       sender      => 'youremail@yourcompany.com',
   ```

```
    recipients  => 'recipientemail@company.com',
    subject     => 'Tables are being modified during maintenance period!',
    message     => msg);

END email_alert;
```

Replace `youremail@yourcompany.com` with your e-mail address, and `recipientemail@company` with the e-mail address of the person you want to receive the notification.

3. Grant `EXECUTE` permissions on this procedure to `DVSYS`.

   ```
   GRANT EXECUTE ON email_alert TO DVSYS;
   ```

Step 3: Create an Oracle Database Vault Rule Set That Uses the E-mail Security Alert

Follow these steps:

1. As the `DV_OWNER` account, start Oracle Database Vault Administrator.

   See "Starting Oracle Database Vault Administrator" on page 3-1 for more information.

2. In Oracle Database Vault Administrator, create a rule that enforces the

3. In the Administration page, under Database Vault Feature Administration, click **Rule Sets**.

4. In the Rule Sets page, click **Create**.

5. Create the new rule set using the following settings:

   - **Name**: Cannot Modify Tables During Maintenance Period
   - **Description**: Rules to prevent tables from being created, altered, or dropped during the maintenance period
   - **Status**: Enabled
   - **Evaluation Options**: Any True
   - **Audit Options**: Audit On Failure
   - **Error Handling Options**:
     - **Custom Event Handler Option**: Execute On Failure
     - **Custom Event Handler Logic**: `dbvacctmgr.email_alert`

6. Click **OK**.

7. In Rule Sets page, select the Cannot Modify Tables During Maintenance Period rule set and then click **Edit**.

8. In the Edit Rule Set page, under Rules Associated To The Rule Set, click **Create**.

9. In the Create Rule page, enter the following settings:

   - **Name**: Maintenance Window
   - **Rule Expression**: Enter a setting to indicate a period of time for the maintenance window. A realistic time range would be the following:

     ```
     TO_CHAR(SYSDATE,'HH24') BETWEEN '22' AND '23'
     ```

     This time enforces the rule set at between 10 p.m. and 11 p.m. To test the rule set, you can enter a more convenient time, for example, between 10 a.m. and 10:59 a.m.:

```
TO_CHAR(SYSDATE,'HH24') BETWEEN '10' AND '11'
```

10. Click **OK**.

Step 4: Test the E-mail Security Alert

Follow these steps:

1. Log on to SQL*Plus as a regular user.

   For example:

   ```
   sqlplus scott
   Enter password: password
   ```

   If the SCOTT account is locked and expired, you can unlock this account and create a new password as follows:

   ```
   ALTER USER SCOTT ACCOUNT UNLOCK IDENTIFIED BY password;
   ```

2. Create a dummy table.

   ```
   CREATE TABLE my_test
   (col1 varchar2);
   ```

3. During the period when the Cannot Modify Tables During Maintenance Period rule set takes place, try running dropping the my_test:

   ```
   DROP TABLE my_test;
   ERROR at line 1:
   ORA-01031: insufficient privileges
   ```

   As you can see, SCOTT cannot drop this table, even though he owns it.

   As a result, you should receive an e-mail with the following message:

   ```
   Tables are being modified during maintenance period!
   ```

4. Outside the period when the Cannot Modify Tables During Maintenance Period rule set is in effect, for example, at 11:15 a.m., drop the my_test table:

   ```
   DROP TABLE my_test;
   Table dropped.
   ```

You can use rule sets with realms, command rules, factors, and secure application roles. For example, to restrict a DROP TABLE statement on any table in the BIZAPP schema to execute only on the corporate network between 10:00 p.m. and 10:59 p.m., you could create a command rule that uses the Cannot Modify Tables During Maintenance Period rule set.

With this command rule in place, a database administrator can be prevented from intentionally or accidentally destroying data assets outside the security policy constraints.

Example 5–1 illustrates what happens when a database administrator attempts to destroy data outside the security policy constraints.

Example 5–1 Database Administrator Attempting to Destroy Data

```
SQL> CONNECT SYSTEM
Enter password: password
Connected.
SQL> DROP TABLE bizapp.accounts_payable;
```

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

Guidelines for Designing Rule Sets

Follow these 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.
- Leverage 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 "Example of How Rule Sets Work" on page 5-9 for an example.

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.

You can check system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Database Control, which is installed by default with Oracle Database), Statspack, and TKPROF. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Database Control, refer to its online Help. *Oracle Database Performance Tuning Guide* describes the Statspack and TKPROF utilities.

Related Reports

Table 5–1 lists Oracle Database Vault reports that are useful for analyzing rule sets and the rules within them. See Chapter 11, "Oracle Database Vault Reports" for information about how to run these reports.

Table 5–1 Reports Related to Rule Sets

Report	Purpose
"Rule Set Configuration Issues Report" on page 11-4	To find rule sets that have no rules defined or enabled
"Secure Application Configuration Issues Report" on page 11-4	To find secure application roles that have incomplete or disabled rule sets
"Command Rule Configuration Issues Report" on page 11-3	To find rule sets that are incomplete or disabled

6 Configuring Command Rules

This chapter describes how to create and configure command rules. It includes the following sections:

- What Are Command Rules?
- Commands with Default Rules
- SQL Statements That Can Be Protected by Command Rules
- Creating and Editing a Command Rule
- Deleting a Command Rule
- How Command Rules Work
- Example of How Command Rules Work
- Guidelines for Configuring Command Rules for SQL Statements
- How Command Rules Affect Performance
- Related Reports

What Are 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 executes 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 "Defining Realm Authorization" on page 4-5.

A command rule has the following attributes, in addition to its bonding operations and authorization functionality:

- SQL statement the command rule will protect
- Owner of the object the command rule will affect
- Database object the command rule will affect
- Whether the command rule is enabled or not
- 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 6-3.

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 authorized and a command rule violation is created. Chapter 5, "Configuring Rule Sets" describes rule sets in detail.

You can define a command rule for a CONNECT event that can determine whether a session is allowed after the normal authentication process, Oracle Label Security initialization, factor initialization, and the Oracle Label Security integration complete. In addition, you can disable or enable a command rule when necessary, and apply the same rule to realms and command rules.

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" on page 6-7 for more information.

This chapter explains how to configure command rules by using Oracle Database Vault Administrator. To configure command rules by using the PL/SQL interfaces and packages provided by Oracle Database Vault, refer to the following appendixes:

- Appendix H, "PL/SQL Interfaces to Oracle Database Vault"
- Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package"

Commands with Default Rules

Table 6–1 lists default command rules that Oracle Database Vault provides.

Table 6–1 Default Command Rules

SQL Statement	Object Name	Rule Set Name
ALTER PROFILE	-	Can Maintain Accounts/Profiles
ALTER SYSTEM	-	Allow System Parameters
ALTER USER	-	Can Maintain Own Account
CREATE PROFILE	-	Can Maintain Accounts/Profiles
CREATE USER	-	Can Maintain Accounts/Profiles
DROP PROFILE	-	Can Maintain Accounts/Profiles
DROP USER	-	Can Maintain Accounts/Profiles
GRANT	SYS.DBMS_RLS package	Can Grant VPD Administration

Table 6–1 (Cont.) Default Command Rules

SQL Statement	Object Name	Rule Set Name
REVOKE	SYS.DBMS_RLS package	Can Grant VPD Administration

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.

The following default command rules on an Oracle Virtual Private Database (VPD) prevent the database administrator from giving VPD capabilities to an account.

- GRANT
- REVOKE

Only the accounts with the DV_OWNER role can use the GRANT and REVOKE statements pertaining to the SYS.DBMS_RLS object and the EXECUTE privilege.

SQL Statements That Can Be Protected by Command Rules

You can protect the following SQL statements by using command rules:

ALTER CLUSTER

ALTER DIMENSION

ALTER FUNCTION

ALTER INDEX

ALTER INDEXTYPE

ALTER JAVA

ALTER OPERATOR

ALTER OUTLINE

ALTER PACKAGE

ALTER PACKAGE BODY

ALTER PROCEDURE

ALTER PROFILE

ALTER RESOURCE COST

ALTER ROLE

ALTER ROLLBACK SEGMENT

CREATE CONTEXT

CREATE DATABASE LINK

CREATE DIMENSION

CREATE DIRECTORY

CREATE FUNCTION

CREATE INDEX

CREATE INDEXTYPE

CREATE JAVA

CREATE LIBRARY

CREATE OPERATOR

CREATE OUTLINE

CREATE PACKAGE

CREATE PACKAGE BODY

CREATE PROCEDURE

CREATE PROFILE

DROP FUNCTION

DROP INDEX

DROP INDEXTYPE

DROP JAVA

DROP LIBRARY

DROP OPERATOR

DROP OUTLINE

DROP PACKAGE

DROP PACKAGE BODY

DROP PROCEDURE

DROP PROFILE

DROP ROLE

DROP ROLLBACK SEGMENT

DROP SEQUENCE

DROP SNAPSHOT

```
ALTER SEQUENCE
ALTER SNAPSHOT
ALTER SNAPSHOT LOG
ALTER SYNONYM
ALTER SYSTEM
ALTER TABLE
ALTER TABLESPACE
ALTER TRIGGER
ALTER TYPE
ALTER TYPE BODY
ALTER USER
ALTER VIEW
ANALYZE CLUSTER
ANALYZE INDEX
ANALYZE TABLE
ASSOCIATE STATISTICS
AUDIT
COMMENT
COMMIT
CONNECT
CREATE CLUSTER
CREATE ROLE
CREATE ROLLBACK SEGMENT
CREATE SCHEMA
CREATE SEQUENCE
CREATE SNAPSHOT
CREATE SNAPSHOT LOG
CREATE SYNONYM
CREATE TABLE
CREATE TABLESPACE
CREATE TRIGGER
CREATE TYPE
CREATE TYPE BODY
CREATE USER
CREATE VIEW
DELETE
DISASSOCIATE STATISTICS
DROP CLUSTER
DROP CONTEXT
DROP DATABASE LINK
DROP DIMENSION
DROP DIRECTORY
DROP SNAPSHOT LOG
DROP SYNONYM
DROP TABLE
DROP TABLESPACE
DROP TRIGGER
DROP TYPE
DROP TYPE BODY
DROP USER
DROP VIEW
EXECUTE
GRANT
INSERT
NOAUDIT
RENAME
REVOKE
SELECT
TRUNCATE CLUSTER
TRUNCATE TABLE
UPDATE
```

Creating and Editing a Command Rule

Follow these steps:

1. Log in to Oracle Database Vault Administrator using a database account that has been granted the DV_OWNER role.

   At a minimum, you must have the DV_ADMIN role. "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

2. In the Administration page, under Database Vault Feature Administration, click **Command Rules**.

3. In the Command Rules page:

   - To create a new command rule, click **Create**.
   - To edit an existing command rule, select it from the list and then click **Edit**.

4. In the Create (or Edit) Command Rule page, enter the following settings, and then click **OK**.

   - General
   - Applicability
   - Rule Set

General

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. The default is **Enabled**. This attribute is mandatory.

Applicability

Enter the following settings:

- **Object Owner**: From the list, select the owner of the object the command rule will affect. You can use wildcard characters such as %. (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 6-3 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.
- **Object Name**: Enter the name of the database object that the command rule will affect, or specify % to select all database objects. However, you cannot use wildcard characters with text, for example, `EMP_%` to specify all tables beginning with the characters `EMP_`. 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" on page 6-7 for more information.

Rule Set

From the list, select the rule set that you want to associate with the command rule. This attribute is mandatory.

If the rule set evaluates to true, then the SQL statement succeeds. If it evaluates to false, the statement fails, and then Oracle Database Vault creates a command rule violation. (You can track such rule violations by using the Command Rule Configuration Issues Report, discussed in Chapter 11.) Any auditing and custom event handling associated with the rule set occurs as a part of the command rule processing.

See Chapter 5, "Configuring Rule Sets" for more information about rule sets.

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. See "Oracle Database Vault Public Views" on page D-10 for more information.

To delete a command rule:

1. In the Oracle Database Vault Administration page, select **Command Rules**.
2. In the Command Rules page, select the command rule that you want to remove.
3. Click **Remove**.
4. In the Confirmation page, click **Yes**.

How Command Rules Work

"How Realms Work" on page 4-7 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 a command rule is executed:

1. Oracle Database Vault queries for all rules that the account is attempting to use.

   For SELECT, DDL, and DML statements, more than one command rule 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. You can disable (or set the **Disabled** setting for) either a command or a rule set of a command. If a command is disabled, then the command is not allowed to run regardless of its associated rule set. That is, the command itself is disabled. However, if you disable a rule set of a command, then the rule set is not checked. In that case, the command is allowed to run without any condition. If the command has been set to the **Disabled** setting, then you are prevented from querying the table even though you have the SELECT privilege on the table. (Only the rule set has been disabled, not the command itself.)

Example of How Command Rules Work

Suppose you want to allow only the accounts with a role called FIN_DBA to create new tables. You could create a command rule with the following settings:

- **Command**: Enter CREATE TABLE
- **Object Owner**: Enter % to select all object owners.
- **Object Name**: Enter % to select all objects.
- **Rule Set**: Enter Can Create Tables

  The Can Create Tables rule set includes a rule that checks to see if the account has the FIN_DBA role.

If a database account with CREATE TABLE privileges issues the CREATE TABLE statement but does not have the FIN_DBA role, then Oracle Database Vault prevents the account from creating the new table and generates an error.

```
SQL> CREATE TABLE accounts_western (
 acct_num NUMBER(10),
 acct_name VARCHAR(10));

ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-20400: Command Rule Violation for CREATE TABLE
ORA-06512: at "DVSYS.AUTHORIZE_EVENT", line 35
```

```
ORA-06512: at line 13
```

See also "Example of How Rule Sets Work" on page 5-9 for another example of how a command rule can work with a rule set.

Guidelines for Configuring Command Rules for SQL Statements

Follow these guidelines to configure 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 schemas, design the command rule to stop the SELECT statement on those specific schemas, rather than creating a general command rule to prevent SELECT statements in all cases.

- When designing rules for the CONNECT event, be careful to include logic that does not inadvertently lock out the Oracle Database Vault Owner or Administrator.

  If the account has been locked out, you can disable Oracle Database Vault, unlock the account, and then reenable Oracle Database Vault. See Appendix B, "Enabling and Disabling Oracle Database Vault" for more information.

- Sometimes you need to 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.

- 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 second 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 Database Control, which is installed by default with Oracle Database), Statspack, and TKPROF. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Database Control, refer to its online Help. *Oracle Database Performance Tuning Guide* describes the Statspack and TKPROF utilities.

Related Reports

Table 6–2 lists Oracle Database Vault reports that are useful for analyzing command rules. See Chapter 11, "Oracle Database Vault Reports" for information about how to run these reports.

Table 6–2 Reports Related to Command Rules

Report	Purpose
"Command Rule Audit Report" on page 11-5	To find audit records generated by command rule processing operations
"Command Rule Configuration Issues Report" on page 11-3	To track rule violations, in addition to other configuration issues the command rule may have
"Object Privilege Reports" on page 11-6	To find object privileges that the command rule affects
"Sensitive Objects Reports" on page 11-7	To find objects that the command rule affects
"Rule Set Configuration Issues Report" on page 11-4	To find rules sets that have no rules defined or enabled, which may affect the command rules that use them

7

Configuring Factors

This chapter describes how to create and configure factors. It includes the following sections:

- What Are Factors?
- Default Factors
- Creating a Factor
- Editing a Factor
- Adding an Identity to a Factor
- Deleting a Factor
- How Factors Work
- Example of How Factors Work
- Guidelines for Designing Factors
- How Factors Affect Performance
- Related Reports

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. "Default Factors" on page 7-2 describes the default factors in detail. 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 described in "Oracle Database Vault PL/SQL Factor Functions" on page H-5 are factor-specific functions that you can use in rule expressions.

Factors have values (identities) and are further categorized by their factor types. "Factor Identification" on page 7-5 explains more about factor identities. See "Factor Type" under "General" on page 7-4 for information about factor types.

You also can integrate factors with Oracle Label Security labels. "Integrating Oracle Database Vault with Oracle Label Security" on page 9-2 explains how. See "Example of Integrating Oracle Database Vault with Oracle Label Security" on page 9-5 for more information.

You can run reports on the factors that you create in Oracle Database Vault. See "Related Reports" on page 7-19 for more information.

This chapter explains how to configure factors by using Oracle Database Vault Administrator. To configure factors by using the PL/SQL interfaces and packages provided by Oracle Database Vault, refer to the following appendixes:

- Appendix H, "PL/SQL Interfaces to Oracle Database Vault"
- Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package"

Default Factors

Oracle Database Vault provides a set of default factors. You can create custom factors based on these default factors using your own PL/SQL retrieval methods. See "Oracle Database Vault PL/SQL Factor Functions" on page H-5 for a listing of the PL/SQL functions for these factors.

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.)

- **Authentication_Method**: Returns 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**: Defines the IP address and retrieval method for a client to the database server.
- **Database_Domain**: Defines the domain of the database as specified in the `DB_DOMAIN` initialization parameter.
- **Database_Hostname**: Defines the host name and retrieval method for a database.
- **Database_Instance**: Defines the instance identifier and retrieval method for a database instance.
- **Database_IP**: Defines the IP address and retrieval method for a database server.
- **Database_Name**: Defines the name of the database as specified in the `DB_NAME` initialization parameter.
- **Domain**: Defines 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**: Returns enterprise-wide identity for the user:
  - For enterprise users: the Oracle Internet Directory-distinquished 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/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**: Returns 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**: Returns the ISO abbreviation for the language name, a shorter form than the existing `LANGUAGE` parameter.
- **Language**: Returns 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**: Returns the name of the computer that was used for the current session. If you need to find out whether the computer was used for a client or server session, you can compare this setting with the Database_Hostname factor to make the determination.
- **Network_Protocol**: Returns the network protocol being used for communication, as specified in the `PROTOCOL=protocol` portion of the connect string.
- **Proxy_Enterprise_Identity**: Returns the Oracle Internet Directory DN when the proxy user is an enterprise user.

- **Proxy_User**: Returns the name of the database user who opened the current session on behalf of SESSION_USER.
- **Session_User**: Returns 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. "Guidelines for Designing Factors" on page 7-18 provides advice on designing factors.

To create a factor:

1. Log in to Oracle Database Vault Administrator using a database account that has been granted the DV_OWNER role.

   At a minimum, you must have the DV_ADMIN role. "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

2. In the Administration page, under Database Vault Feature Administration, click **Factors**.
3. In the Factors page, click **Create**.
4. In the Create Factor page, enter the following settings, and then click **OK**:
   - General
   - Factor Identification
   - Evaluation
   - Factor Labeling
   - Retrieval Method
   - Validation Method
   - Assignment Rule Set
   - Audit Options
   - Error Options

General

In the General area, enter the following information:

- **Name**: Enter a name up to 30 characters in mixed-case, without spaces. Oracle Database Vault will create 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 Network, Oracle Database Vault creates the DVF.F$NETWORK function. This attribute is mandatory.

  "Oracle Database Vault PL/SQL Factor Functions" on page H-5 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. (See "Default Factors" on page 7-2 for more information.) 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 geography.

> **Note:** To create user-defined factor types by using the Oracle Database Vault `DVSYS.DBMS_MACADM` package, use the `CREATE_FACTOR_TYPE` procedure, described in "CREATE_FACTOR_TYPE Function" on page E-26.

Factor Identification

Under Factor Identification, select how to resolve the identity of a factor. This attribute is mandatory. The values are as follows:

- **By Method** (default): 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 March 23, 2007, the **By Method** option would return the following value:

  ```
  2007-04-23
  ```

- **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 "Mapping an Identity" on page 7-12 for more information about mapping identities.

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" on page 7-19 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.

> **See Also:** "Adding an Identity to a Factor" on page 7-10 for more information about factor identities

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:

```
SQL> CONNECT macadmin/
Enter password: password
SQL> SELECT dvf.f$database_ip FROM dual;
F$DATABASE_IP
-------------------------------------------------------------
172.16.0.3
```

You can also use the DVSYS.GET_FACTOR function to determine the identity of a factor that is made available for public access. For example:

```
SQL> SELECT get_factor('DATABASE_IP') FROM dual;
GET_FACTOR('DATABASE_IP')
-------------------------------------------------------------
172.16.0.3
```

Evaluation

Under Evaluation, select how you want the factor to be evaluated and assigned an identity. See "How Factors Affect Performance" on page 7-18 for the performance effect of session factors. This attribute is mandatory.

The values are as follows:

- **By Session** (default): Evaluates the factor when a database session is created.
- **By Access**: Evaluates the factor each time it is accessed (say, referenced by an application) as well as when the database session is first created.

Factor Labeling

Under Factor Labeling, 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 9-2 for information on integrating OLS labels with a factors.

The values are as follows:

- **By Self** (default): 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, Oracle Database Vault merges the labels by using the Oracle Label Security Algorithm page 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.

Retrieval Method

Under Retrieval Method, enter a PL/SQL expression that retrieves the identity of a factor or a constant. It can use up to 255 characters in mixed-case. 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 PL/SQL Factor Functions" on page H-5
- "Factor Functions Within DVSYS.DBMS_MACADM" on page E-20
- Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package"

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'))
```

See also the default factors provided with Oracle Database Vault for examples of retrieval methods. "Default Factors" on page 7-2 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
```

Validation Method

Under Validation Method, 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, you must provide `DVSYS` with the `GRANT EXECUTE` privilege on the object.

Write the function using one 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 the following sections for factor-specific and general utility functions that you can use to build the validation method:

- "Oracle Database Vault PL/SQL Factor Functions" on page H-5
- "Factor Functions Within DVSYS.DBMS_MACADM" on page E-20
- Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package"

Assignment Rule Set

Under Assignment Rule Set, select a rule set from the list 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 5, "Configuring Rule Sets" explains how to create rule sets.

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:

```
DVSYS.SET_FACTOR('GEO_STATE','VIRGINIA');
```

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 7-16 for more information.

Audit Options

Under Audit Options, select from the settings to generate a custom Oracle Database Vault audit record. You can use the Factor Audit Report to display the generated audit records. (See "Related Reports" on page 7-19 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. This attribute is mandatory.

The values are as follows:

- **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 an Identity" on page 7-10 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.

Error Options

Under Error Options, select from the following to specify the processing that occurs when a factory identity cannot be resolved. This attribute is mandatory.

The values are as follows:

- **Show Error Message** (default): 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.

> **See Also:**
>
> - "Editing a Factor" on page 7-9
> - "Adding an Identity to a Factor" on page 7-10

Editing a Factor

To edit a factor:

1. In the Oracle Database Vault Administration page, select **Factors**.
2. In the Factors page, select the factor that you want to edit.
3. Click **Edit**.
4. Modify the factor as necessary, and then click **OK**.

See Also:

- "Creating a Factor" on page 7-4 to modify the settings created for a new factor
- "Adding an Identity to a Factor" on page 7-10 to add or modify an identity for the factor

Adding an Identity to a Factor

After you create a new factor, you can add an identity to it. An identity is the actual value of the factor. For example, the identity of an IP_Address factor could be the IP address of 234.43.41.99.

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 7-4. 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 "How Factors Work" on page 7-14 for more information about how a factor behaves during a database session.

Creating and Configuring an Identity

To create and configure an identity:

1. In the Oracle Database Vault Administration page, select **Factors**.
2. In the Factors page, select the factor to which you want to add the identity.
3. Click **Edit**.
4. In the Edit Factor page, scroll down to Identities and click **Create**.
5. In the Create Identity page, enter the following settings and then click **OK**:
   - General
   - Label Identity

General

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)

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.

Example 7–1 shows examples of running these two functions.

Example 7–1 Determining the Trust Level of a Factor Identity

```
FUNCTION get_trust_level(p_factor IN VARCHAR2) RETURN NUMBER

FUNCTION get_trust_level_for_identity(p_factor IN VARCHAR2, p_identity IN
VARCHAR2) RETURN NUMBER
```

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 table to find trust levels for the Network factor greater than or equal to 5:

```
SQL> SELECT value, trust_level from dvsys.dba_dv_identity
 WHERE trust_level >= 5
 AND factor_name='Network'

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 Appendix H, "PL/SQL Interfaces to Oracle Database Vault" for more information about the Oracle Database Vault functions.

Label Identity

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.

To label an identity:

1. In the Create Identity page, under Label Identity, select the OLS label from the **Available OLS Labels** 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*.

   > **Note:** You can select multiple labels by holding down the Ctrl key as you click each label that is to be selected.

2. Click **Move** to move the OLS label to the **Selected OLS Labels** list.
3. Repeat Step 1 and Step 2 to select more OLS labels.

   You can select only one label for each OLS policy.

4. Click **OK** to finish labeling the identity.

Editing a Factor Identity

To edit a factor identity:

1. In the Edit Factor page, scroll down to Identities and select the identity you want to edit.
2. Click **Edit**.
3. In the Edit Identity page, modify the identity as necessary.
4. Click **OK**.

Deleting a Factor Identity

Before you delete a factor identity, you can locate the various references to it by querying the factor-related Oracle Database Vault views. See "Oracle Database Vault Public Views" on page D-10 for more information.

To delete a factor identity:

1. In the Edit Factor page, scroll down to Identities and select the identity you want to remove.
2. Click **Remove**.
3. In the Confirmation page, click **Yes**.

Mapping an Identity

After you create, edit, and save the 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, see "Identity Configuration Issues Report" on page 11-3.

To map an identity to a factor:

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

   "Creating a Factor" on page 7-4 describes how to create factors.

2. For the parent factor, create a new factor identity.

   "Creating and Configuring an Identity" on page 7-10 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 Factors page, select the parent factor from the Factors page and then click **Edit**.

   b. In the Edit Factor page, under Identities, select the parent factor identity and then click **Edit**.

   c. In the Edit Identity page, click **Create** under Map Identity.

   d. In the Create Identity Map page, select a factor name from the **Contributing Factor** box.

      This is the child factor to which you want to map the parent factor.

   e. Select a **Map Condition**.

      This setting lets you select an operator to compare the contributing (child) factor values.

   f. Enter a value for the Low Value and High Value (optional) fields.

      For example, consider a scenario where the Contributing Factor to the Factor Network is set to Client_IP, the **Map Condition** is set to `Between`, the **Low Value** is set to 192.168.0.1 and the **High Value** is set to 192.168.0.254. This means that whenever the client IP address lies in the specified address range of 192.168.0.1 to 192.168.0.254, the parent factor evaluates to a predefined identity, say, INTRANET.

   g. Click **OK** to map the parent factor-identity to the child factor-identity.

      You can map different identities of a parent factor to different identities of the contributing factor. For example, the INTRANET identity maps to an IP address range of 192.168.0.1 to 192.168.0.254. The REMOTE identity can map to an IP address range that excludes the address range 192.168.0.1 to 192.168.0.254.

      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).

   h. Repeat Step c to Step g 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 192.168.0.1 and 192.168.0.254. So, if an authorized accounting financial application program, running on a client with IP address 192.168.0.23 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 "Oracle Database Vault Public Views" on page D-10 for more information.

To delete a factor:

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 **Remove**.
5. In the Confirmation page, click **Yes**.

How Factors Work

The following topics in this section explain how Oracle Database Vault processes factors:

- 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

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 will be 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, follow these steps:

1. Disable Oracle Database Vault.

   Appendix B explains how to disable and reenable Oracle Database Vault.
2. Delete the CONNECT command rules by using the Oracle Database Vault DVSYS.DBMS_MACADM.DELETE_COMMAND_RULE procedure, for example:

   ```
   EXECUTE DBMS_MACADM.DELETE_COMMAND_RULE('CONNECT', '%', '%');
   ```

   (You still can make Oracle Database Vault API calls when Oracle Database Vault has been disabled.)
3. Reenable Oracle Database Vault.

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.

Example 7–2 shows an example of using the DVSYS.GET_FACTOR function.

Example 7–2 Using DVSYS.GET_FACTOR to Retrieve a Factor

```
SQL> 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 PL/SQL Factor Functions" on page H-5 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 7-14.

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 7-14, 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 7-14 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.

Example of How Factors Work

Oracle Database Vault provides a set of default factors that can be used for routine tasks such as finding host names, IP addresses, names of database instances, and names of session users (See "Default Factors" on page 7-2.) However, you can create custom factors if you want. For example, you can create a Program factor that returns the name of the program that is running in the current session. To create this factor, the security administrator used the following settings:

- **Factor Type**: Application

This setting was selected because the Program factor is used to identify the name of the program that established the current session.

- **Factor Identification**: By method

  The method the factor uses is defined under **Retrieval Method**.

- **Evaluation**: By Session

  This setting was selected so that the factor can be evaluated when the database session begins.

- **Factor Labeling**: By Self

  This setting labels the identities directly, rather than merging the labels of the identities for the factors that form an identity map.

- **Retrieval Method**: `DVSYS.DBMS_MACADM.GET_SESSION_INFO('PROGRAM')`

  "Oracle Database Vault PL/SQL Factor Functions" on page H-5 provides more information about all the functions available for factors, including `GET_SESSION_INFO`.

- **Validation Method**: None specified

  No validation methods are needed. The retrieval method performs all necessary functions.

- **Assignment Rule Set**: None selected

  No rule set is needed for this type of factor.

- **Audit Options**: Always, and then select **Trust Level Less Than Zero**.

- **Error Options**: Show Error Message

  When you call `DVSYS.GET_FACTOR` and if there is an error, you could use this setting to reveal any configuration problems. (See "Oracle Database Vault Run-Time PL/SQL Procedures and Functions" on page H-1 for more information about `GET_FACTOR`.)

- **Identities**: None created.

  No identities are needed for this type of factor.

When the Program factor was created, it was immediately available for use in database sessions and applications using standard Oracle SQL and PL/SQL expressions. You can query this factor by using standard SQL or by using it in PL/SQL expressions, because it has been exposed as an Oracle function to all database sessions.

Example 7–3 shows how to query the current value of the Program factor using an account with the `DV_OWNER` role

Example 7–3 Using GET_SESSION_INFO to Query the Value of a Factor

```
sqlplus macsys
Enter password: password
SQL> SELECT DVSYS.DBMS_MACADM.GET_SESSION_INFO('PROGRAM') FROM dual;
F$PROGRAM
---------------------
sqlplus@stadb38 (TNS V1-V3)
```

This example showed how a fairly simple factor works. For a more complex example on how to integrate an Oracle Database Vault factor with an Oracle Label Security label, see "Example of Integrating Oracle Database Vault with Oracle Label Security" on page 9-5.

Guidelines for Designing Factors

Follow these 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 Performance 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 9-2 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

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 17 default factors listed in "Default Factors" on page 7-2 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, the other factors consume resources that could otherwise be used elsewhere. In this case, you should remove the uneccessary 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 Database Control, which is installed by default with Oracle Database), `Statspack`, and `TKPROF`. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Database Control, refer to its online Help. *Oracle Database Performance Tuning Guide* describes the `Statspack` and `TKPROF` utilities.

Related Reports

Table 7–1 lists Oracle Database Vault reports that are useful for analyzing factors and their identities. See Chapter 11, "Oracle Database Vault Reports" for information about how to run these reports.

Table 7–1 Reports Related to Factors and Their Identities

Report	Purpose
"Factor Audit Report" on page 11-5	To audit factors, for example, to find factors that failed to be evaluated
"Factor Configuration Issues Report" on page 11-3	To find configuration issues, such as disabled or incomplete rule sets, or to audit issues that may affect the factor
"Factor Without Identities Report" on page 11-3	To find factors that have had no identities assigned yet
"Identity Configuration Issues Report" on page 11-3	To find factors that have invalid label identities or no map for the identity
"Rule Set Configuration Issues Report" on page 11-4	To find rule sets that have no rules defined or enabled, which may affect the factors that use them

8

Configuring Secure Application Roles for Oracle Database Vault

This chapter describes how to configure secure application roles. It includes the following sections:

- What Are Secure Application Roles for Oracle Database Vault?
- Creating and Editing Secure Application Roles
- Securing a Secure Application Role
- Deleting a Secure Application Role
- How Secure Application Roles Work
- Example of How Secure Application Roles Work
- How Secure Application Roles Affect Performance
- Related Reports

What Are Secure Application Roles for 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.

You can run reports on secure application roles that you create in Oracle Database Vault. See "Related Reports" on page 8-6 for more information.

This chapter explains how to configure secure application roles by using Oracle Database Vault Administrator. To configure secure application roles by using the PL/SQL interfaces and packages provided by Oracle Database Vault, refer to the following appendixes:

- Appendix H, "PL/SQL Interfaces to Oracle Database Vault"
- Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package"

Creating and Editing Secure Application Roles

Follow these steps:

1. Log in to Oracle Database Vault Administrator using a database account that has been granted the DV_OWNER role.

   At a minimum, you must have the DV_ADMIN role. "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

2. In the Administration page, under Database Vault Feature Administration, click **Secure Application Roles**.

3. In the Secure Application Roles page:

   - To create a new secure application role, click **Create**.
   - To edit an existing secure application role, select it from the list and then click **Edit**.

     Remember that 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:

     ```
     SQL> 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 DVSYS.DBMS_MACSEC_ROLES.SET_ROLE in your application code to accomplish this. See "SET_ROLE Function" on page F-2 for more information about the SET_ROLE function.

4. In the Create (or Edit) Role page, enter the following settings and then click **OK**.

   - General
   - Rule Set

General

Enter the following settings:

- **Role**: Enter the name using no more than 30 characters, with no spaces. Preferably, enter the role name in upper case letters, though you are not required to do so. 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. The default is **Enabled**. This attribute is mandatory.
  - **Enabled**: Calls the DVSYS.DBMS_MACSEC_ROLES.SET_ROLE function to determine whether or not a role is set for a database session.

    See "SET_ROLE Function" on page F-2 for more information about this function.
  - **Disabled**: Prevents the need for the DVSYS.DBMS_MACSEC_ROLES.SET_ROLE function.

See "Oracle Database Vault PL/SQL Packages" on page H-8 for more information about the DVSYS.DBMS_MACSEC_ROLES.SET_ROLE function.

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 DVSYS.DBMS_MACSEC_ROLES.SET_ROLE, if the rule set evaluates to true, then Oracle Database Vault sets the role for the database session. If the rule set evaluates to false, then the role is not set.

See Chapter 5, "Configuring Rule Sets" for more information about rule sets.

Securing a Secure Application Role

Users who have database administrative privileges can use the DROP ROLE command to delete secure application roles that were created using Oracle Database Vault.

To prevent the database administrator from deleting a secure application role, when you create secure application roles, protect them by using a realm. To do so, add the role to a realm authorization. See "Defining Realm Authorization" on page 4-5 for more information.

Deleting a Secure Application Role

Before you delete a secure application role, you can locate the various references to it by querying the role-related Oracle Database Vault views. See "Oracle Database Vault Public Views" on page D-10 for more information.

To delete a secure application role:

1. Check and modify any applications that may be using the secure application role that you want to delete.
2. In the Oracle Database Vault Administration page, select **Secure Application Roles**.
3. In the Secure Application Roles page, select the role that you want to remove.
4. Click **Remove**.
5. In the Confirmation page, click **Yes**.

How Secure Application Roles Work

The process flow for a secure application role that is managed by Oracle Database Vault is as follows:

1. Create or update the role either in Oracle Database Vault Administrator or by using the secure application role-specific functions in the DVSYS.DBMS_MACADM package.

   See "Secure Application Role Functions Within DVSYS.DBMS_MACADM" on page E-34 for more information.

2. Modify your application to call the role, by using the DVSYS.DBMS_MACSEC_ROLES.SET_ROLE function.

   See "SET_ROLE Function" on page F-2 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.

Example of How Secure Application Roles Work

Suppose you wanted to create a secure application role to allow employees to manage their own accounts within the in-house applications your company has created. To do so, you would follow these steps:

- Step 1: Create a Rule Set to Be Used with the Secure Application Role
- Step 2: Create the Secure Application Role Using the Rule Set
- Step 3: Grant Privileges to the Role
- Step 4: Enable the Role in Your Applications
- Step 5: Test the New Secure Application Role

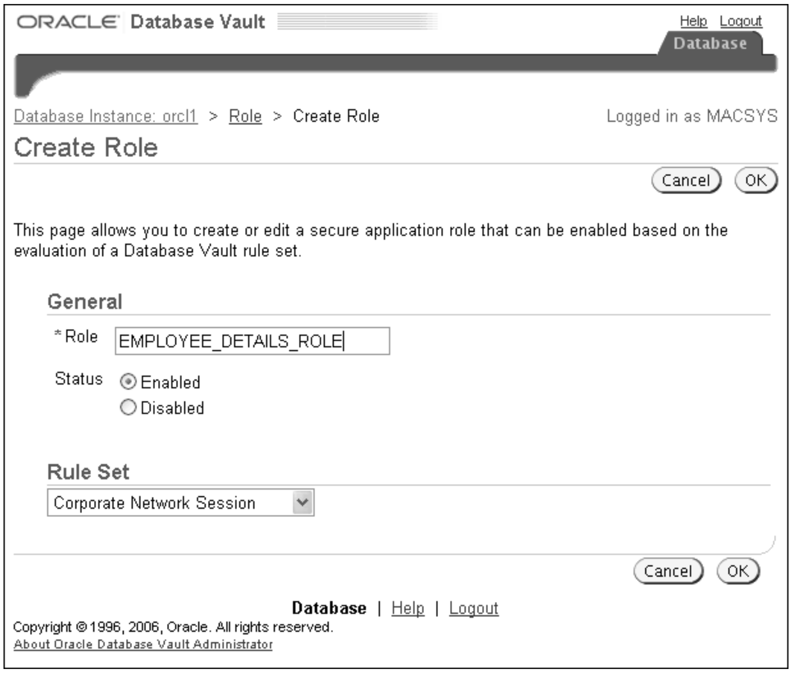
Step 1: Create a Rule Set to Be Used with the Secure Application Role

You can use an existing rule set or create a new one. See Chapter 5, "Configuring Rule Sets" for information about creating a new rule set. Oracle Database Vault provides a set of default rule sets that you can use.

Step 2: Create the Secure Application Role Using the Rule Set

Figure 8–1 shows how to create a secure application role for employees to manage their own accounts.

Figure 8–1 Secure Application Role Example



Create 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.

General

* Role: EMPLOYEE_DETAILS_ROLE

Status: ● Enabled ○ Disabled

Rule Set

Corporate Network Session

Copyright © 1996, 2006, Oracle. All rights reserved.

About Oracle Database Vault Administrator

Step 3: Grant Privileges to the Role

Next, connect as a database administrator who can grant the SELECT privilege and is a realm owner of the realm protecting the object. Grant SELECT privileges for the data to the employee_details_role as follows.

```
SQL> CONNECT appadmin
Enter password: password
Connected.

SQL> GRANT SELECT ON hr.employees TO employee_details_role;
Grant succeeded.
```

Step 4: Enable the Role in Your Applications

After the secure application role has been created, applications or database sessions can enable it by calling the DVSYS.DBMS_MACSEC_ROLES.SET_ROLE function:

```
DVSYS.DBMS_MACSEC_ROLES.SET_ROLE('employee_details_role')
```

Step 5: Test the New Secure Application Role

Now, with these settings in place, a powerful system administrator can be prevented from remotely accessing any sensitive employee data, such as the salary for file clerk Ida Neau. Here is what happens when the system administrator tries this from the office:

```
SQL> CONNECT system@hrdb_from_cubicle
Enter password: password
Connected.

SQL> select dvf.f$network from dual
CORPORATE

SQL> EXEC DVSYS.DBMS_MACSEC_ROLES.SET_ROLE('employee_details_role');
PL/SQL procedure successfully completed.

SQL> SELECT count(*) from hr.employees
COUNT(*)
----------
        57
----------
```

But here is what happens when the administrator tries to set the employee_details_role role from home:

```
SQL> CONNECT system@hrdb_from_home
Enter password: password
Connected.

SQL> select dvf.f$network from dual
REMOTE

SQL> EXEC DVSYS.DBMS_MACSEC_ROLES.SET_ROLE('employee_details_role');
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

As you can see, the security protection in this example happens at the time the administrator tries to set the SET_ROLE call—not when he or she attempts data access.

Had the administrator first tried accessing the data in HR.EMPLOYEES, the same thing would happen: the security protection in place would prevent data access.

How Secure Application Roles Affect Performance

You can check system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Database Control, which is installed by default with Oracle Database), Statspack, and TKPROF. For more information about Oracle Enterprise Manager, see the Oracle Enterprise Manager documentation set. For information about Database Control, refer to its online Help. *Oracle Database Performance Tuning Guide* describes the Statspack and TKPROF utilities.

Related Reports

Table 8–1 lists Oracle Database Vault reports that are useful for analyzing secure application roles. See Chapter 11, "Oracle Database Vault Reports" for information about how to run these reports.

Table 8–1 Reports Related to Secure Application Roles

Report	Purpose
"Secure Application Role Audit Report" on page 11-5	To find 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 11-4	To find secure application roles that have nonexistent database roles, or incomplete or disabled rule sets
"Rule Set Configuration Issues Report" on page 11-4	To find 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 11-9	To find information about powerful database accounts and roles

9 Integrating Oracle Database Vault with Other Oracle Products

This chapter explains how you can integrate Oracle Database Vault with the following Oracle products:

- Integrating Oracle Database Vault with Enterprise User Security
- Integrating 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 Enterprise User Security

You can integrate Oracle Database Vault with Oracle 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 roles as needed, you can create a rule set for the Enterprise users to allow or disallow their access.

To configure an Enterprise User authorization:

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 5-5 to create a new rule. In the Create Rule page, enter the following PL/SQL in the Rule Expression field:

   ```
   SYS_CONTEXT('USERENV','EXTERNAL_NAME') = 'user_domain_name'
   ```

   Replace *user_domain_name* with the domain, for example:

   ```
   SYS_CONTEXT('USERENV','EXTERNAL_NAME') = 'myserver.us.mycompany.com'
   ```

2. Add this rule to a new rule set.

   "Creating a Rule Set" on page 5-2 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 database that you want to protect.

"Defining Realm Authorization" on page 4-5 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.

For more information about Enterprise User Security, see *Oracle Database Enterprise User Security Administrator's Guide*.

Integrating Oracle Database Vault with Transparent Data Encryption

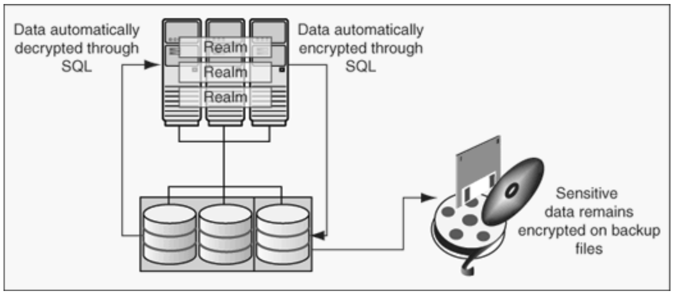
Oracle Database Vault works with Transparent Data Encryption (TDE). With Transparent Data Encryption, an application administrator can use a single one line command to alter a table and encrypt a column. Subsequent inserts into that table column will be written to disk encrypted transparent to the SQL. This means that no SQL modification, database triggers, or views are required.

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.

So, if you have TDE enabled, Oracle Database Vault will work with it seemlessly and without any additional configuration. TDE also can be enabled in an Oracle Database Vault environment with any additional configuration.

Figure 9–1 shows how Oracle Database Vault realms handle encrypted data.

Figure 9–1 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. To do so, define a policy predicate that is a PL/SQL function or expression. Then, for each function or expression, you can use the `DVF.F$` PL/SQL function that is created for each factor.

Integrating Oracle Database Vault with Oracle Label Security

This section includes the following topics:

- How Oracle Database Vault Is Integrated with Oracle Label Security
- Requirements for Using Oracle Database Vault with Oracle Label Security
- Using an Oracle Database Vault Factor with an Oracle Label Security Policy
- Example of Integrating Oracle Database Vault with Oracle Label Security
- Related Reports

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. "Example of Integrating Oracle Database Vault with Oracle Label Security" on page 9-5 provides an example of how to accomplish this type of integration.

You can audit the integration with Oracle Label Security by using the Label Security Integration Audit Report. See "Label Security Integration Audit Report" on page 11-5 for more information.

You can use the Oracle Database Vault APIs to integrate Oracle Database Vault with Oracle Label Security. See Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package" for more information.

For more information about Oracle Label Security labels, levels, and policies, see *Oracle Label Security Administrator's Guide*.

You can run reports on the Oracle Database Vault and Oracle Label Security integration. See "Related Reports" on page 9-8 for more information.

Requirements for Using Oracle Database Vault with Oracle Label Security

You must have the following requirements in place before you use Oracle Database Vault with Oracle Label Security:

- Oracle Label Security is licensed separately. Make sure you have purchased a license to use it.
- Before you install Oracle Database Vault, you must have already installed Oracle Label Security.
- Ensure that you have the appropriate Oracle Label Security policies defined. For more information, see *Oracle Label Security Administrator's Guide*.

Using an Oracle Database Vault Factor with an Oracle Label Security Policy

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.

Use the following steps to define factors that contribute to the maximum allowable data label of an Oracle Label Security policy:

1. Log in to Oracle Database Vault Administrator using a database account that has been granted the `DV_OWNER` role.

   At a minimum, you must have the `DV_ADMIN` role. "Starting Oracle Database Vault Administrator" on page 3-1 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 "Defining Realm Authorization" on page 4-5 for more information.

3. In the Administration page, under Database Vault Feature Administration, click **Label Security Integration**.

4. In the Label Security Policies page:

   - To register a new label security policy, click **Create**.
   - To edit an existing label security policy, select it from the list and then click **Edit**.

5. Enter the following settings and then click **OK**:

   - General
   - Label Security Policy Factors

General

Under General, 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 need to 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 need to 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 DVSYS.DBMS_MACADM package to specify a merge algorithm, see Table E–62, " Merge Algorithm Codes" on page E-37 for a full listing of possible merge algorithms.

- **Label for Initialization Errors**: Optionally enter a label for initialization errors. The label specified for initialization errors is set when a configuration error or run-time error occurs during session initialization. You can use this setting to assign the session a data label that prevents access or updates to any data the policy protects until the issue is resolved.

Label Security Policy Factors

To select a factor to associate with an Oracle Label Security policy:

1. In the **Available Factors** list under Label Security Policy Factors, select the factor that you want to associate with the Oracle Label Security policy.
2. Click **Move** to move the factor to the **Selected Factors** list.

> **Note:** You can select multiple factors by holding down the **Ctrl** key as you click each factor that you want to select.

After you associate a factor with an Oracle Label Security policy, you can label the factor identities using the labels for the policy. "Adding an Identity to a Factor" on page 7-10 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.

Example of Integrating Oracle Database Vault with Oracle Label Security

You can use Oracle Database Vault factors in conjunction 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.

To demonstrate how you can integrate Oracle Database Vault with Oracle Label Security, assume that you must create the following access controls for your company:

- Allow access to sensitive accounting data to database sessions that originate from the corporate Intranet
- Prevent access to this data to database sessions that originate over the corporate VPN network

To do so, you can create a factor named `Network`, whose identity values are Intranet and Remote. These values are resolved based on values of a second factor, Client_IP, which stores the IP address of the computer making the connection. Because of the interdependency of the two factors, the Network factor is considered a parent factor and the Client_IP factor is its child factor. You establish this interdependency by creating an identity map. Then finally, you associate an Oracle Label Security label with the `Intranet` and `Remote` identities in `Network`.

To accomplish this, follow these steps:

- Step 1: Create the Network Factor
- Step 2: Create Identity Maps for the Network Intranet and Remote Identities
- Step 3: Associate the Network Factor with an Oracle Label Security Policy
- Step 4: Test the Configuration

Step 1: Create the Network Factor

First, create the Network factor:

1. In the Oracle Database Vault Administrator home page, select **Factors**.
2. In the Factors page, select **Create**.
3. In the Create Factors page, enter the following settings:
   - **Name**: `Network`
   - **Description**: Example factor for application access through network.
   - **Factor Type**: `Physical`
   - **Factor Identification**: `By Factors`
   - **Evaluation**: `For Session`.
   - **Factor Labeling**: `By Self`
   - **Retrieval Method**: Leave this field blank.
   - **Validation Method**: Leave this field blank.
4. Click **OK**.

   The Network factor appears in the Factors page listing. Now you are ready to create its identities, Intranet and Remote.
5. In the Factors page, select **Network** and then click **Edit**.
6. In the Edit Factor page, go to the Identities section and click **Create**.
7. In the Create Identity page, enter the following settings:
   - **Value**: `Intranet`
   - **Trust Level**: Select **Very Trusted**.
   - **Label Identity**: `EMPLOYEE_POLICY - HIGHLY_SENSITIVE`
8. Click **OK**.
9. Repeat Step 6 and Step 7 to create the Remote identity, using the following settings:
   - **Value**: `Remote`
   - **Trust Level**: Select **Trusted**.
   - **Label Identity**: `EMPLOYEE_POLICY - SENSITIVE`
10. Click **OK**.

You do not need to create the Client_IP factor; it is supplied with Oracle Database Vault.

Step 2: Create Identity Maps for the Network Intranet and Remote Identities

Next, you are ready to create an identity map for each of the Network factor Intranet and Remote identities. This map links the Network and Client_IP factors, by identifying the Network as the parent factor and Client_IP as its child factor.

First, create the identity map for the Intranet identity:

1. In the Factors page in Oracle Database Vault Administrator, select **Network** and then click **Edit**.
2. In the Edit Factor page, under Identities, select **Intranet** and then click **Edit**.
3. In the Edit Identity page, under Map Identity, click **Create**.
4. In the Create Identity Map page, enter the following settings:
   - **Contributing Factor**: `Client_IP`
   - **Map Condition**: Select `Like`. Then enter the following values:
     - **Low Value**: `192.168.10%`
     - **High Value**: Leave this field blank.
5. Click **OK**.
6. Create a second mapping by repeating Step 3 and Step 4, using the following settings:
   - **Contributing Factor**: `Client_IP`
   - **Map Condition**: Select `Is Null`. Then enter the following values:
     - **Low Value**: `NULL`
     - **High Value**: Leave this field blank.

   The second mapping handles situations in which a user logs in from the database host computer, the Client_IP factor will be null.

The settings should appear as follows:

Table 9–1 Identify Map Settings

Child Factor Name	Operation Value	Operand 1	Operand 2
Client_IP	Equal	Is Null	Blank
Client_IP	Like	192.168.10%	Blank

Next, create an identity map for the Remote identity by using the following settings:

- **Contributing Factor**: `Client_IP`
- **Map Condition**: Select `Like`. Then enter the following values:
  - **Low Value**: `192.168.67%`
  - **High Value**: Leave this field blank.

Once you have completed this mapping, you can check the values for the Network factor by executing the following statement in SQL*Plus from a corporate and remote/VPN network connection or session:

```
SQL> SELECT dvf.f$network FROM dual
---------------------------------------------
Remote
```

Step 3: Associate the Network Factor with an Oracle Label Security Policy

Finally, you are ready to associate the Network factor with an Oracle Label Security policy.

Follow these steps:

1. In the Administration page, under Database Vault Feature Administration, click **Label Security Integration**.
2. In the Label Security Policies page, click **Create**.
3. In the Create Label Security Policy page, enter the following settings:
   - **Label Security Policy**: Select **EMPLOYEE_POLICY**.
   - **Algorithm**: Select LII - Minimum Level/Intersection/Intersection.
   - **Label for Initialization Errors**: Leave this setting blank.
   - **Label Security Policy Factors**: Select **Network** and click **Move** to move it to the Selected Factors list.
4. Click **OK**.

Step 4: Test the Configuration

To test this configuration, try logging on from both the local Intranet connection and a remote connection.

First, try the connection from the local Intranet connection. As you can see, you have access to all the records in `HR.EMPLOYEES`:

```
SQL> SELECT COUNT(*) FROM HR.EMPLOYEES;
----------------------------------------------------------------------------
350

SQL> SELECT DVF.F$NETWORK FROM DUAL;

F$NETWORK
----------------------------------------------------------------------------
Intranet
```

Now try it from a remote connection. In this case, you have access only to a limited subset (200 rows) of the `HR.EMPLOYEES` table records:

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

COUNT(*)
----------------------------------------------------------------------------
200

SQL> SELECT DVF.F$NETWORK FROM DUAL;

F$NETWORK
----------------------------------------------------------------------------
Remote
```

Related Reports

Table 9–2 lists Oracle Database Vault reports that are useful for analyzing the Oracle Database Vault and Oracle Label Security integration. See Chapter 11, "Oracle Database Vault Reports" for information about how to run these reports.

Table 9–2 Reports Related to Database Vault and Oracle Label Security Integration

Report	Purpose
"Factor Configuration Issues Report" on page 11-3	To find factors in which the Oracle Label Security policy does not exist.
"Identity Configuration Issues Report" on page 11-3	To find any invalid label identities (the Oracle Label Security label for this identity has been removed and no longer exists).
"Security Policy Exemption Report" on page 11-10	To find 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.

10

Monitoring Oracle Database Vault

This chapter explains how you can monitor Oracle Database Vault. It includes the following sections:

- Security Violation Attempts
- Database Configuration and Structural Changes
- Security Policy Changes by Category
- Security Policy Changes Detail

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.

Follow these steps:

1. Log in to Oracle Database Vault Administrator with an account that uses the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

   "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

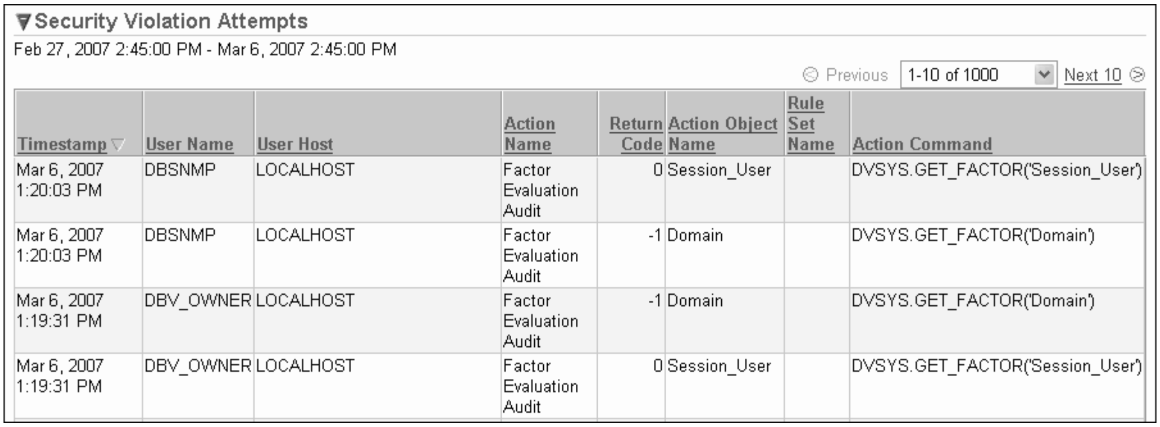
2. In the Administration page, click **Monitor**.

3. At the top of the Monitor page, set a period of time for the monitoring action by selecting from the **Show Records For** list and clicking **Go**.

   This section of the Monitor page also indicates the last time the data on the page was refreshed.

4. In the Monitor page, click **Security Violation Attempts**.

   A table appears, listing security policy changes.



Security Violation Attempts

Feb 27, 2007 2:45:00 PM - Mar 6, 2007 2:45:00 PM

Timestamp ▽	User Name	User Host	Action Name	Return Code	Action Object Name	Rule Set Name	Action Command
Mar 6, 2007 1:20:03 PM	DBSNMP	LOCALHOST	Factor Evaluation Audit	0	Session_User		DVSYS.GET_FACTOR('Session_User')
Mar 6, 2007 1:20:03 PM	DBSNMP	LOCALHOST	Factor Evaluation Audit	-1	Domain		DVSYS.GET_FACTOR('Domain')
Mar 6, 2007 1:19:31 PM	DBV_OWNER	LOCALHOST	Factor Evaluation Audit	-1	Domain		DVSYS.GET_FACTOR('Domain')
Mar 6, 2007 1:19:31 PM	DBV_OWNER	LOCALHOST	Factor Evaluation Audit	0	Session_User		DVSYS.GET_FACTOR('Session_User')

Database Configuration and Structural Changes

You can view structural changes to the database or database schema objects. This feature also audits statements such as CREATE TABLE, ALTER TABLE, DROP TABLE, and ALTER DATABASE. It audits all commands, not just commands that are used in command rules. For example, if someone has unexpectedly altered a table on a production system, you can use this feature to determine what is happening.

Follow these steps:

1. Log in to Oracle Database Vault Administrator with an account that uses the DV_OWNER, DV_ADMIN, or DV_SECANALYST role.

   "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

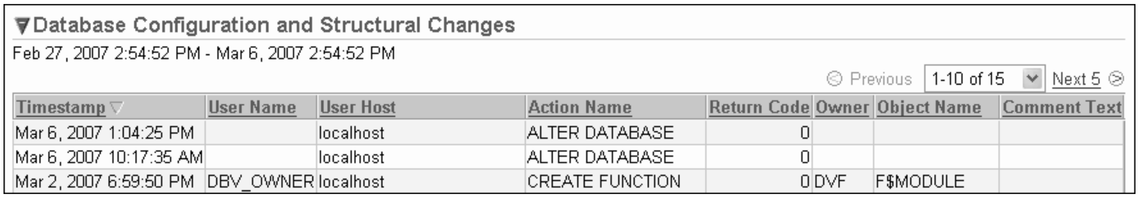
2. In the Administration page, click **Monitor**.

3. At the top of the Monitor page, set a period of time for the monitoring action by selecting from the **Show Records For** list and clicking **Go**.

   This section of the Monitor page also indicates the last time the data on the page was refreshed.

4. In the Monitor page, click **Database Configuration and Structural Changes**.

   A table similar to the following appears:



Database Configuration and Structural Changes

Feb 27, 2007 2:54:52 PM - Mar 6, 2007 2:54:52 PM

Timestamp ▽	User Name	User Host	Action Name	Return Code	Owner	Object Name	Comment Text
Mar 6, 2007 1:04:25 PM		localhost	ALTER DATABASE	0			
Mar 6, 2007 10:17:35 AM		localhost	ALTER DATABASE	0			
Mar 2, 2007 6:59:50 PM	DBV_OWNER	localhost	CREATE FUNCTION	0	DVF	F$MODULE	

Security Policy Changes by Category

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.

- **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. Before you can audit the database audit policy, the `AUDIT_TRAIL` initialization parameter must be set to `DB`. See "About the Baseline Oracle Database Vault Auditing Policy" on page A-1 for an example.
- **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.

To monitor security policy changes by category:

1. Log in to Oracle Database Vault Administrator using the Oracle Database Vault owner (with the `DV_OWNER` role) or security analyst account (with the `DV_SECANALYST` role).

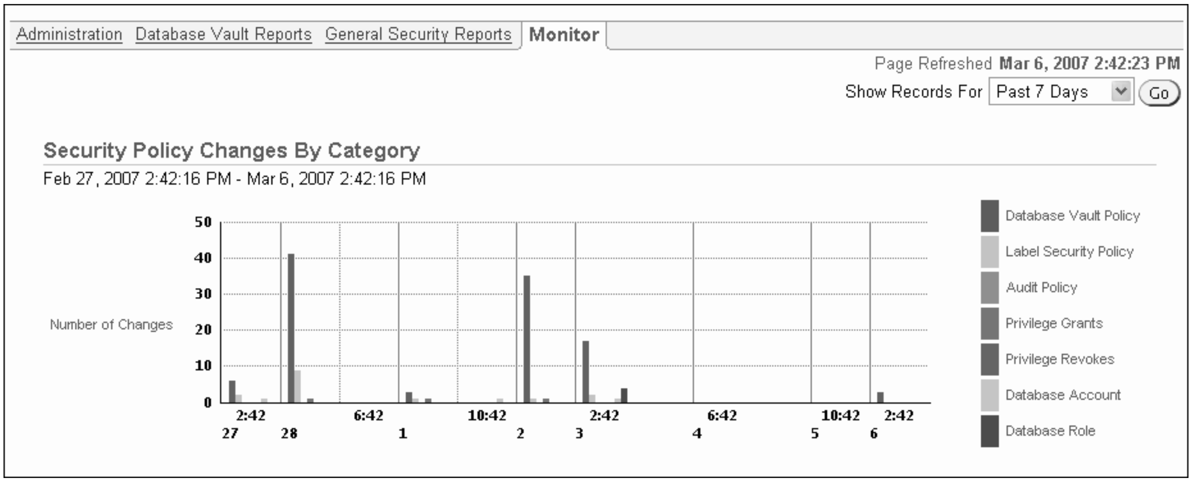
   "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log on.

2. In the Administration page, click **Monitor**.
3. At the top of the Monitor page, set a period of time for the monitoring action by selecting from the **Show Records For** list and clicking **Go**.

   This section of the Monitor page also indicates the last time the data on the page was refreshed.

4. In the Monitor page, check the graph under **Security Policy Changes by Category**.

   A graph similar to the following appears, which shows the number of security policy changes based on the following categories: Oracle Database Vault policy, Oracle Label Security policy, audit policy, privilege grants and revokes, database accounts, and database roles.



Page Refreshed **Mar 6, 2007 2:42:23 PM**

Show Records For Past 7 Days Go

Security Policy Changes By Category

Feb 27, 2007 2:42:16 PM - Mar 6, 2007 2:42:16 PM

Security Policy Changes Detail

You can check the details of security policy changes, such the user who made the change, the action that occurred, the time stamp of the change, and so on.

To monitor security policy changes by detail:

1. Log in to Oracle Database Vault Administrator with an account that uses the DV_OWNER, DV_ADMIN, or DV_SECANALYST role.

   "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

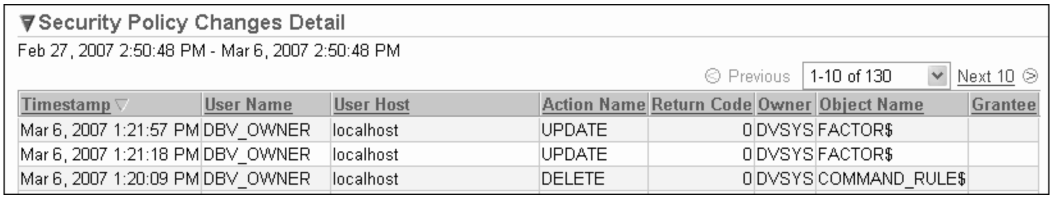
2. In the Administration page, click **Monitor**.

3. At the top of the Monitor page, set a period of time for the monitoring action by selecting from the **Show Records For** list and clicking **Go**.

   This section of the page also indicates the last time the data on the page was refreshed.

4. In the Monitor page, click **Security Policy Changes by Detail**.

   A table appears, listing the details for security policy changes.



▼Security Policy Changes Detail

Feb 27, 2007 2:50:48 PM - Mar 6, 2007 2:50:48 PM

Timestamp ▽	User Name	User Host	Action Name	Return Code	Owner	Object Name	Grantee
Mar 6, 2007 1:21:57 PM	DBV_OWNER	localhost	UPDATE	0	DVSYS	FACTOR$	
Mar 6, 2007 1:21:18 PM	DBV_OWNER	localhost	UPDATE	0	DVSYS	FACTOR$	
Mar 6, 2007 1:20:09 PM	DBV_OWNER	localhost	DELETE	0	DVSYS	COMMAND_RULE$	

11

Oracle Database Vault Reports

This chapter explains how you can run reports on various activities in Oracle Database Vault. It includes the following sections:

- Categories of Oracle Database Vault Reports
- Who Can Run the Oracle Database Vault Reports?
- How to Run Oracle Database Vault Reports
- Generating Oracle Database Vault Reports
- Generating General Security Reports

Categories of 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. 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. For more information about these roles, see the following sections:

- "Oracle Database Vault Owner Role, DV_OWNER" on page D-4
- "Oracle Database Vault Configuration Administrator Role, DV_ADMIN" on page D-5
- "Oracle Database Vault Security Analyst Role, DV_SECANALYST" on page D-6

How to Run Oracle Database Vault Reports

To run Oracle Database Vault reports:

1. Log in to Database Vault Administrator.

   "Starting Oracle Database Vault Administrator" on page 3-1 explains how to log in.

   Users who have the following roles can run the reports:

   - DV_OWNER
   - DV_ADMIN
   - DV_SECANALYST (least privileged)

2. Select either **Database Vault Reports** or **General Security Reports**.

   These report categories are described in the following sections:

   - "Generating Oracle Database Vault Reports" on page 11-2
   - "Generating General Security Reports" on page 11-5

3. Select a report and click **Run Report** to run the report.

   You can run many of the reports without any input parameters. For example, if you select the Audit Privileges Report, and click **Run Report**, then you can immediately see the report results. However, some of the available reports require at least one input parameter before the results can be displayed.

   The Report Results page displays the report content in a tabular fashion with the column headings shown at the top of the report. The page displays the report title as well as the date and time when the report was run. Click **Return to Reports Menu** to return to the Reports page, so that you can select and run a different report if you want.

   Some of the reports require at least one input parameter to be provided before they can be run. For example, when you select **Object Dependencies Report** and click **Run Report**, the Report Parameters page is displayed. The **Owner** box enables you to select the database account that owns the object. The **Object Name** field specifies the name of the object. You can use wildcard characters like the percentage sign (%), which defaults to all object names. The **Result Set Size** parameter determines the maximum number of result rows that are displayed. If you want all records to be displayed, then select **All**.

   The parameters that you enter on this page are passed directly to the SQL query that generates the report results. Click **Run Report** to display the report results based on the specified parameters.

Generating Oracle Database Vault Reports

To generate Oracle Database Vault reports, click the **Database Vault Reports** tab, and then select from the following categories of reports:

- Oracle Database Vault Configuration Issues Reports
- Oracle Database Vault Auditing Reports

Oracle Database Vault Configuration Issues Reports

The configuration issues reports are:

- Command Rule Configuration Issues Report
- Factor Configuration Issues Report
- Factor Without Identities Report
- Identity Configuration Issues Report
- Realm Authorization Configuration Issues Report
- Rule Set Configuration Issues Report
- Secure Application Configuration Issues Report

Command Rule Configuration Issues Report

The Command Rule Configuration Issues Report displays command rules for which the following configuration issues exist:

- 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.

Factor Configuration Issues Report

The Factor Configuration Issues Report displays Oracle Database Vault factors for which the following configuration issues exist:

- 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 the following configuration issues exist:

- Label identity for the Oracle Label Security label for this identity has been removed and no longer exists.
- No map exists for the identity.

Realm Authorization Configuration Issues Report

The Realm Authorization Configuration Issues Report displays Oracle Database Vault realm information where the following configuration issues exist.

- 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.

Rule Set Configuration Issues Report

The Rule Set Configuration Issues Report displays Oracle Database Vault rule set information where the following configuration issue exists:

No rules are defined or enabled for a rule set.

Secure Application Configuration Issues Report

The Secure Application Configuration Issues Report displays Database Vault secure application role information where the following configuration issues exist:

- Database role does not exist. This can happen when the database role has been dropped.
- Rule set for role is disabled.
- Rule set for role is incomplete.

Oracle Database Vault Auditing Reports

The auditing reports are:

- 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

Timestamp

Return Code

Account

User Host

Instance Number

Object Name

Rule Set

Command

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.

Generating General Security Reports

To generate general security reports, click the **General Security Reports** tab, and then select from the following reports:

- 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 are:

- 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. It 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
DMSYS
DVSYS
EXFSYS
LBACSYS
MDSYS
ORDSYS
PUBLIC
SYS
SYSMAN
SYSTEM
WKSYS
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. It 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 as well as 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 are:

- 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. It 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 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 are:

- 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 execute privileges on system packages that can be used to access operating system resources or other powerful system packages. The following system packages are included:

DBMS_ALERT

DBMS_BACKUP_RESTORE

DBMS_CAPTURE_ADM

DBMS_DDL

DBMS_DISTRIBUTED_TRUST_ADMIN

DBMS_FGA

DBMS_JOB

DBMS_LDAP

DBMS_LOB

DBMS_LOGMNR

DBMS_LOGMNR_D

DBMS_OBFUSCATION_TOOLKIT

DBMS_ORACLE_TRACE_AGENT

DBMS_PIPE

DBMS_RANDOM

DBMS_REPAIR

DBMS_REPCAT

DBMS_REPCAT_ADMIN

DBMS_RESOURCE_MANAGER

DBMS_RESOURCE_MANAGER_PRIVS

DBMS_RLS

DBMS_SESSION

DEBUG_EXTPROC

UTL_FILE

UTL_HTTP

UTL_SMTP

UTL_TCP

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. It includes the following system tables and views:

ALL_SOURCE

ALL_USERS

APPROLE$

AUD$

AUDIT_TRAIL$

DBA_ROLE_PRIVS

DBA_ROLES

DBA_TAB_PRIVS

DBMS_BACKUP_RESTORE

DEFROLE$

FGA_LOG$

LINK$

OBJ$

OBJAUTH$

OBJPRIV$

PROFILE$

PROXY_ROLE_DATA$

PROXY_ROLE_INFO$

ROLE_ROLE_PRIVS

SOURCE$

STATS$SQLTEXT

STATS$SQL_SUMMARY

STREAMS$_PRIVILEGED_USER

SYSTEM_PRIVILEGE_MAP

TABLE_PRIVILEGE_MAP

TRIGGER$

USER$

USER_HISTORY$

USER_TAB_PRIVS

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 can be used to determine as to 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. It 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 are:

- Privileges Distribution By Grantee Report
- Privileges Distribution By Grantee, Owner Report
- Privileges Distribution By Grantee, Owner, Privilege Report

Privileges Distribution By Grantee Report

The Privileges Distribution By Grantee Report displays the count of privileges granted to a database account or role. This 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 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 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 are:

- 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

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 I, "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, "Oracle Database Vault Auditing Policies" 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. This is a very powerful system privilege: it enables the `IMPORT_FULL_DATABASE` and `EXPORT_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 I-6.

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 following roles granted to them:

- 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.

Initialization Parameters and Profiles Reports

The initialization parameters and profiles reports are:

- 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 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 of time 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 are:

- 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. The 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 "About the Baseline Oracle Database Vault Auditing Policy" on page A-1, as well as 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`. See "About the Baseline Oracle Database Vault Auditing Policy" on page A-1 for an example of how to set this parameter. For more information about the AUDIT_TRAIL parameter, see *Oracle Database SQL Language Reference*.

Other Security Vulnerability Reports

The other security vulnerability reports are:

- 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. It 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 or not 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. 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

Oracle Database Vault Auditing Policies

This appendix includes the following sections:

- About the Baseline Oracle Database Vault Auditing Policy
- Enabling Oracle Database Vault Auditing
- Oracle Database Vault Audit Policy Settings
- Oracle Database Vault Specific Audit Events
- Archiving the Oracle Database Vault Audit Trail

About the Baseline Oracle Database Vault Auditing Policy

A baseline auditing policy is installed with Oracle Database Vault. This policy audits the following components:

- Access control configuration information that is stored in the oracle Database Vault database tables
- Information that is stored in the Oracle Catalog (rollback segments, tablespaces, and so on)
- The use of system privileges
- The Oracle Label Security configuration

Enabling Oracle Database Vault Auditing

Before you can capture the Oracle Database Vault audit information, you must enable the audit trail by setting the `AUDIT_TRAIL` initialization parameter to `OS`, which writes the audit trail record to an operating system file. The default setting for `AUDIT_TRAIL` is `DB`, which writes the audit trail record to the data dictionary. For security reasons, use `OS` instead, because it reduces the likelihood of a Denial of Service (DoS) attack. To set this parameter, log on to SQL*Plus with `SYSDBA` privileges, set `AUDIT_TRAIL` to `OS`, and then restart the database. For example:

```
sqlplus "sys / as sysdba"
Enter password: password
Connected.

SQL> ALTER SYSTEM SET AUDIT_TRAIL = OS SCOPE=SPFILE;
SQL> ALTER SYSTEM SET AUDIT_TRAIL_DEST = directory_path DEFERRED;
SQL> SHUTDOWN NORMAL;
SQL> STARTUP;
```

For more information about the `AUDIT_TRAIL` parameter, see *Oracle Database Security Guide* and *Oracle Database Reference*.

Oracle Database Vault Audit Policy Settings

Table A–1 shows the audit settings in the core database auditing policy used with Oracle Database Vault.

Table A–1 Database Vault Audit Policy Settings

Audit Setting Type	Audited Commands (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`

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–1 (Cont.) Database Vault Audit Policy Settings

Audit Setting Type	Audited Commands (BY ACCESS and on Success or Failure Unless Otherwise Noted)
User Audit Settings for DVSYS/DVF User Audit Settings for LBACSYS See Table D–2, " Database Accounts Used by Oracle Database Vault" on page D-7 for more information about these accounts. See also these sections for detailed information on the DVSYS and DVF schemas: ■ "DVSYS Schema" on page D-1 ■ "DVF Schema" on page D-2	ADMINISTER DATABASE TRIGGER
	ALTER *object*
	AUDIT SYSTEM
	BECOME USER
	CLUSTER
	COMMENT
	CONTEXT
	CREATE *object*
	DATABASE LINK
	DEBUG
	DIRECTORY
	DROP *object*
	EXECUTE LIBRARY (whenever not successful)
	EXECUTE PROCEDURE (whenever not successful)
	EXEMPT ACCESS POLICY
	EXEMPT IDENTITY 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)

Table A–1 (Cont.) Database Vault Audit Policy Settings

Audit Setting Type	Audited Commands (BY ACCESS and on Success or Failure Unless Otherwise Noted)

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`
Object Audit Settings for `LBACSYS`
	`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`
	`INSERT TABLE/VIEW`
	`RENAME PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/VIEW/TABLE`
	`SELECT SEQUENCE/TABLE/VIEW` (whenever not successful)
	`UPDATE TABLE/VIEW`

Oracle Database Vault Specific Audit Events

After you have enabled auditing, you can define a database audit policy for auditing system commands, users, objects, and so on. However, the database audit policy does not inherently support several Oracle Database Vault events.

Oracle Database Vault defines custom events that you can choose to audit. This enables you to audit events not protected by the database audit policy. For example, if the run-time access control processing of retrieving the identifier for a factor fails, the audit options for the factor may dictate that this event be audited.

Custom Audit Events

The following list describes some of the custom audit events:

- **Session Initialization Failed Audit**: Audits instances where the access control session fails to initialize.
- **Command Rule Audit**: Audits the rule set processing results. You can audit both successful and failed processing.
- **Realm Authorization Audit**: Realm authorizations can be managed using rule sets. You can audit the rule set processing results.
- **Realm Violation Audit**: A realm violation occurs when the database account, performing an action on a realm object, is not authorized to perform that action in the realm. You can audit realm violations.
- **Factor Assignment Audit**: A factor can have an associated rule set that is used to assign an identity to the factor at run time. You can audit both successful and failed processing.

- **Factor Evaluation Audit**: Audits instances where a factor identity cannot be resolved and assigned (such as *No data found* or *Too many rows*). You can audit both successful and failed retrievals.
- **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.
- **Secure Role Audit**: Secure application roles can be set based on rule sets. You can audit the associated rule set processing.

See Also:

- "Audit Options" on page 7-8 (for factors)
- "Audit Options" on page 5-3 (for rule sets)
- Defining Realm Authorization in Chapter 4, "Configuring Realms"
- Chapter 11, "Oracle Database Vault Reports" for information about viewing the audit reports

Format of the Oracle Database Vault Audit Trail

The Oracle Database Vault custom audit event records are stored in the `AUDIT_TRAIL$` table, which is part of the `DVSYS` schema. These audit records are not part of the typical Oracle Database audit trail. You can define an archiving policy for this audit trail.

Table A–2 describes the format of the audit trail.

Table A–2 Audit Trail Format

Parameter	Type	Description
`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)
`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$` ■ `FACTOR$`

Table A–2 (Cont.) Audit Trail Format

Parameter	Type	Description
ACTION	NUMBER	Numeric action type code. The corresponding name of the action type is in the ACTION_NAME column. Expected ACTION and ACTION_NAME values are: ■ 10000: Factor Evaluation Audit ■ 10001: Factor Assignment Audit ■ 10002: Factor Expression Audit ■ 10003: Realm Violation Audit ■ 10004: Realm Authorization Audit ■ 10005: Command Authorization Audit ■ 10006: Secure Role Audit ■ 10007: Access Control Session Initialization Audit ■ 10008: Access Control Command Authorization Audit ■ 10009: Oracle Label Security Session Initialization Audit ■ 10010: Oracle Label Security Attempt to Upgrade Label Audit
ACTION_NAME	VARCHAR2(128)	Name of the action type corresponding to the numeric code in the ACTION column. You can extend the audit trail to include your own ACTION_NAME text, based on the audit events passed.
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
SQL_TEXT	VARCHAR2(2000)	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	The unique identifier of the rule that was executing and caused the audit event to trigger
RULE_NAME	VARCHAR2(30)	The unique name of the rule that was executing and caused the audit event to trigger
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	Numeric identifier for each Oracle session

Table A–2 (Cont.) Audit Trail Format

Parameter	Type	Description
STATEMENTID	NUMBER	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	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.
CLIENT_ID	NUMBER	Client identifier for the Oracle session that triggered the audit event.
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

Archiving the Oracle Database Vault Audit Trail

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 the audit trail to prevent it from growing too large.

To archive the Oracle Database Vault audit trail:

1. Ensure that Data Pump Export is installed.

   Log on to SQL*Plus with administrative privileges and then run the following query:

   ```
   sqlplus "sys/as sysdba"
   Enter password: password
   SQL> SELECT ROLE FROM DBA_ROLES WHERE ROLE LIKE '%FULL%'
   ```

   If the query does not return the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles, then Data Pump Export is not installed. To install Data Pump Export, run either the catexp.sql or catalog.sql script. For example:

   ```
   SQL> @/opt/oracle/app/oracle/admin/catexp.sql;
   ```

   See *Oracle Database Utilities* for more information about the Export utility.

2. Disable Oracle Database Vault.

   See "Step 1: Disable Oracle Database Vault" on page B-2 for more information.

3. If the DVSYS account is locked, then unlock it.

   You need to have this account unlocked when you run the EXPDP export command. Ensure that you are still logged on as SYS, connecting AS SYSDBA.

   ```
   ALTER USER dvsys IDENTIFIED BY password ACCOUNT UNLOCK;
   ```

4. Confirm the location of the operating system audit trail.

   ```
   SHOW PARAMETER AUDIT_FILE_DEST

   NAME                TYPE     VALUE
   ------------------- -------  --------------------------------------
   audit_file_dest     string   /opt/oracle/app/oracle/admin/orcl/adump
   ```

   In general, you may want to keep your audit files in one location.

5. In SQL*Plus, create a directory object in which to generate the Oracle Database Vault audit trail. To do so, connect as `SYS` or as any user who has the `CREATE DIRECTORY` privilege.

   For example:

   ```
   CREATE DIRECTORY dv_audit_dir AS '/opt/oracle/app/oracle/admin/orcl/adump';
   ```

   Enclose the directory path in single quotation marks, not double quotation marks.

6. Grant read and write privileges on the directory object to user `DVSYS`.

   For example:

   ```
   GRANT READ, WRITE ON DIRECTORY dv_audit_dir TO dvsys;
   ```

7. At the operating system command line, enter a command similar to the following to export the `DVSYS.AUDIT_TRAIL$` audit table into a new dump file.

   ```
   EXPDP DVSYS/password \
   DIRECTORY=dv_audit_dir \
   TABLES=DVSYS.AUDIT_TRAIL$ \
   QUERY=DVSYS.AUDIT_TRAIL$:"WHERE timestamp < 2007-04-03:19:34:59"
   DUMPFILE=dv_audit_031607.dmp
   ```

   In this specification:

   - `DIRECTORY`: Enter the directory object that you created in Step 5. Ensure that that the user who is running `EXPDP` (`DVSYS` in this example) has read and write permissions on this directory object. If you created the directory object, then you automatically have read and write permissions on it.
   - `TABLES`: Enter `DVSYS.AUDIT_TRAIL$`, the name of the audit trail table.
   - `QUERY`: Optional. This setting writes a subset of the audit table contents to the dump file, in this case, audit records that are less than the `timestamp` column value of `2007-04-03:19:34:59.`
   - `DUMPFILE`: Enter the name of the dump file that you want to create. The default extension is `.dmp`, but you can use any extension. Ensure that the file name you specify is unique.

8. Purge the Oracle Database Vault audit trail table, now that you have archived it.

   For example, if you archived all audit trail records that are less than the `timestamp` column value of 2007-04-03:19:34:59, enter the following statement:

   ```
   DELETE FROM DVSYS.AUDIT_TRAIL$ WHERE timestamp < 2007-04-03:19:34:59;
   ```

   To completely purge the audit trail and remove the extents allocated to the audit trail table, enter the following statement:

   ```
   TRUNCATE TABLE DVSYS.AUDIT_TRAIL$;
   ```

9. Exit SQL*Plus.

B

Enabling and Disabling Oracle Database Vault

You may have 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.

This chapter includes the following sections:

- When You Must Disable Oracle Database Vault
- Step 1: Disable Oracle Database Vault
- Step 2: Perform the Required Tasks
- Step 3: Enable Oracle Database Vault

When You Must Disable Oracle Database Vault

The following situations require you to disable Oracle Database Vault:

- The password for the Oracle Database Vault account manager (with role `DV_ACCTMGR`) has been forgotten.
- The Database Vault Owner (with role `DV_OWNER`) or Database Vault Administrator (with role `DV_ADMIN`) accounts have been inadvertently locked out.
- A rule set associated with the `CONNECT` role has been configured incorrectly. This is resulting in failed database logins for all accounts, including those with the `DV_OWNER` or `DV_ADMIN` role, who could correct this problem.
- You must perform maintenance tasks on Oracle Database Vault.
- You must install any of the Oracle Database optional products, such as Oracle Spatial Data Option or Oracle *inter*Media, by using Database Configuration Assistant (DBCA).
- You are about to install a third-party product, install an Oracle product, or perform an Oracle patch update whose installation may be prevented if Oracle Database Vault is running.
- You need to archive the Oracle Database Vault audit trail.

Step 1: Disable Oracle Database Vault

This section contains the following topics:

- Disabling Oracle Database Vault on UNIX Systems
- Disabling Oracle Database Vault on Windows Systems

Note: After you disable Oracle Database Vault, you still can run the Oracle Database Vault API functions. Note also that after you disable Oracle Database Vault, the ANY privileges are available.

Disabling Oracle Database Vault on UNIX Systems

Follow these steps to disable Oracle Database Vault on UNIX systems:

1. Turn off the software processes. Make sure that the environment variables, ORACLE_HOME, ORACLE_SID, and PATH are correctly set.

   Stop the dbconsole process in case it is running. For both single-instance and Oracle Real Application Clusters installations, use the following command:

   ```
   $ emctl stop dbconsole
   ```

   For single-instance installations, shut down the database instance:

   ```
   $ sqlplus "sys / as sysoper"
   Enter password: password
   SQL> SHUTDOWN IMMEDIATE
   SQL> EXIT
   ```

   For Oracle Real Application Clusters (RAC) installations, shut down each database instance as follows:

   ```
   $ srvctl stop database -d db_name -c "sys/sys_passwd as sysoper"
   ```

   If you cannot connect to the database, then proceed to the next step.

2. Relink the Oracle executable to turn off the Oracle Database Vault option:

   ```
   $ cd $ORACLE_HOME/rdbms/lib
   $ make -f ins_rdbms.mk dv_off
   $ cd $ORACLE_HOME/bin
   $ relink oracle
   ```

   For RAC installations, run these commands on all nodes.

3. Start the database.

   For single-instance database installations:

   ```
   $ sqlplus "sys / as sysoper"
   Enter password: password
   SQL> STARTUP
   SQL> EXIT
   ```

   For RAC installations:

   ```
   $ srvctl start database -d db_name -c "sys/sys_passwd as sysoper"
   ```

4. Run Oracle Database Vault Configuration Assistant (DVCA) by using the dvca -action disable option.

The syntax for dvca -action disable is as follows:

```
dvca -action disable
  -service service_name
  -instance Oracle_instance_name
  -dbname database_name
  -sys_passwd SYS_password
  -owner_account DV_owner_account_name
  -owner_passwd DV_owner_account_password
  [-logfile ./dvca.log]
  [-nodecrypt]
  [-racnode node]
```

In this specification:

- `-action` is the action to perform. In this case the action is disable.
- `-service` is the database service name.
- `-instance` is the name of the database instance.
- `-dbname` is the database name.
- `-sys_passwd` is the `SYS` password. If you use a cleartext password on the command line, you must include the `nodecrypt` option. If you omit the password, DVCA prompts you for it.
- `-owner_account` is the Oracle Database Vault Owner account name.
- `-owner_passwd` is the Oracle Database Vault Owner account password. If you use a cleartext password on the command line, you must include the `nodecrypt` option. If you omit the password, DVCA prompts you for it.
- `-logfile` is an optional flag to specify a log file name and location. You can enter an absolute path, or enter a path that is relative to the location of the *$ORACLE_HOME*`/bin` directory.
- `-silent` is the option to run in command line mode. This option is required if you are not running DVCA in an xterm window.
- `-nodecrypt` is the option to read plaintext passwords.
- `-lockout` is the flag to use to disable `SYSDBA` operating system authentication.

For example:

```
dvca -action disable
  -oh $ORACLE_HOME
  -service myservicename
  -instance myinstance
  -dbname mydbname
  -owner_account myownername
  -logfile dvcalog.txt

Enter SYS password: sys_password
Enter owner password: owner_password
```

Disabling Oracle Database Vault on Windows Systems

Follow these steps to disable Oracle Database Vault on Windows systems:

1. Stop the database service.

   In the Control Panel, under Administrative Services, select the **Services** utility. Select the **Standard** tab, right-click the following services, and from the menu, select **Stop**:

   - **OracleService*SID***
   - **Oracle*HOME*TNSListener**

2. Under *ORACLE_HOME*\bin, rename the oradv10.dll file, for example, oradv10_backup.dll.

3. Restart the database service.

   In the Control Panel, under Administrative Services, select the **Services** utility. Select the **Standard** tab, right-click the following services, and from the menu, select **Start**:

   - **OracleService*SID***
   - **Oracle*HOME*TNSListener**

4. For RAC systems, repeat these steps for each node on which the database is installed.

5. Run Oracle Database Vault Configuration Assistant (DVCA) by using the dvca -action disable option.

   The syntax for dvca -action disable is as follows:

   ```
   dvca -action disable
     -service service_name
     -instance Oracle_instance_name
     -dbname database_name
     -sys_passwd SYS_password
     -owner_account DV_owner_account_name
     -owner_passwd DV_owner_account_password
     [-logfile ./dvca.log]
     [-nodecrypt]
     [-racnode node]
   ```

   In this specification:

   - -action is the action to perform. In this case the action is disable.
   - -service is the database service name.
   - -instance is the name of the database instance.
   - -dbname is the database name.
   - -sys_passwd is the SYS password. If you use a cleartext password on the command line, you must include the nodecrypt option. If you omit the password, DVCA prompts you for it.
   - -owner_account is the Oracle Database Vault Owner account name.
   - -owner_passwd is the Oracle Database Vault Owner account password. If you use a cleartext password on the command line, you must include the nodecrypt option. If you omit the password, DVCA prompts you for it.

- `-logfile` is an optional flag to specify a log file name and location. You can enter an absolute path, or enter a path that is relative to the location of the *$ORACLE_HOME*/`bin` directory.
- `-silent` is the option to run in command line mode. This option is required if you are not running DVCA in an xterm window.
- `-nodecrypt` is the option to read plaintext passwords.
- `-lockout` is the flag to use to disable `SYSDBA` operating system authentication.

For example:

```
dvca -action disable
  -oh c:\oracle\product\11.1.0\db_1
  -service myservicename
  -instance myinstance
  -dbname mydbname
  -owner_account myownername
  -logfile dvcalog.txt

Enter SYS password: sys_password
Enter owner password: owner_password
```

Step 2: Perform the Required Tasks

With Oracle Database Vault disabled, you can restart your database and perform the following tasks, as required. Advice is as follows:

- If the Oracle Database Vault owner account called `MACSYS` forgets his or her password, for example, you can log in to a database instance as the `SYSTEM` or `SYS` account to create a new password for the Oracle Database Vault owner account as follows

```
$ sqlplus "sys / as sysdba"
Enter password: password
Connected.
SQL> PASSWORD MACSYS
New password: new_password
Retype new password: new_password
```

- Similarly, to unlock a locked account, log in to the database instance as `SYSTEM` or `SYS`, and then unlock the account. For example:

```
SQL> ALTER USER MACSYS ACCOUNT UNLOCK;
```

- To correct a login or connect rule set error, use the `DBMS_MACADM` package or the Oracle Database Vault Administrator interface.

Note: If you are using Oracle Database Vault Administrator, then you must start the `dbconsole` process. You can check the status of the `dbconsole` process by entering the following command from the `$ORACLE_HOME/bin` directory of the Oracle home in which you deployed Database Vault Administrator:

```
./emctl status dbconsole
```

To start `dbconsole`:

```
./emctl start dbconsole
```

- You can perform the installation, upgrade, or other tasks that require security protections to be disabled. If you must run Oracle Database Vault Configuration Assistant (DVCA), ensure that the Oracle Database listener is running. To start the listener, run the following command from the `$ORACLE_HOME/bin` directory:

  ```
  $ ./lsnrctl start
  ```

Step 3: Enable Oracle Database Vault

This section contains the following topics:

- Enabling Oracle Database Vault on UNIX Systems
- Enabling Oracle Database Vault on Windows Systems

Enabling Oracle Database Vault on UNIX Systems

Use the following steps to enable Oracle Database Vault on UNIX systems:

1. Use DVCA to re-enable Oracle Database Vault.

   For example:

   ```
   dvca -action ensable
     -oh $ORACLE_HOME
     -service myservicename
     -instance myinstance
     -dbname mydbname
     -owner_account myownername
     -logfile dvcalog.txt

   Enter SYS password: sys_password
   Enter owner password: owner_password
   ```

   See Step 4 under "Disabling Oracle Database Vault on UNIX Systems" on page B-2 for detailed information about the DVCA syntax.

2. Turn off the software processes. Make sure that the environment variables, `ORACLE_HOME`, `ORACLE_SID`, and `PATH` are correctly set.

   Stop the `dbconsole` process in case it is running. For both single-instance and RAC installations, use the following command:

   ```
   $ emctl stop dbconsole
   ```

3. Shut down the database instance.

   For single-instance installations:

   ```
   $ sqlplus "sys / as sysoper"
   Enter password: password
   Connected.
   SQL> SHUTDOWN IMMEDIATE
   SQL> EXIT
   ```

   For RAC installations:

   ```
   $ srvctl stop database -d db_name -c "sys/sys_passwd as sysoper"
   ```

4. Relink the oracle executable to turn on the Oracle Database Vault option:

   ```
   $ cd $ORACLE_HOME/rdbms/lib
   $ make -f ins_rdbms.mk dv_on
   $ cd $ORACLE_HOME/bin
   ```

```
$ relink oracle
```

For RAC installations, run these commands on all nodes.

5. Start the database:

   For single-instance database installations:

   ```
   $ sqlplus "sys / as sysoper"
   Enter password: password
   Connected.
   SQL> STARTUP
   SQL> EXIT
   ```

   For RAC installations:

   ```
   $ srvctl start database -d db_name -c "sys/sys_passwd as sysoper"
   ```

Enabling Oracle Database Vault on Windows Systems

Follow these steps to enable Oracle Database Vault on Windows systems:

1. Use DVCA to re-enable Oracle Database Vault.

   For example:

   ```
   dvca -action enable
     -oh c:\oracle\product\11.1.0\db_1
     -service myservicename
     -instance myinstance
     -dbname mydbname
     -owner_account myownername
     -logfile dvcalog.txt

   Enter SYS password: sys_password
   Enter owner password: owner_password
   ```

   See Step 5 under "Disabling Oracle Database Vault on Windows Systems" on page B-4 for detailed information about the syntax for DVCA.

2. ///Stop the database service.

   In the Control Panel, under Administrative Services, select the **Services** utility. Select the **Standard** tab, right-click the following services, and from the menu, select **Stop**:

   - **OracleService*SID***
   - **Oracle*HOME*TNSListener**

3. Under *`ORACLE_HOME`*`\bin`, name the backup of the `oradv10.dll` file to its original name.

   For example, if you named it `oradv10_backup.dll`, then name it back to `oradv10.dll`.

4. Restart the database service.

   In the Control Panel, under Administrative Services, select the **Services** utility. Select the **Standard** tab, right-click the following services, and from the menu, select **Start**:

   - **OracleService*SID***
   - **Oracle*HOME*TNSListener**

5. For RAC systems, repeat these steps for each node on which the database is installed.

C

Post-Installation Oracle Database Vault Procedures

This appendix describes the following topics:

- Configuring Oracle Database Vault on RAC Nodes
- Adding Languages to Oracle Database Vault

Configuring Oracle Database Vault on RAC Nodes

After you install Oracle Database Vault for an Oracle Real Application Clusters (RAC) instance, you need to run Oracle Database Vault Configuration Assistant (DVCA) with the `-action optionrac` option on all other RAC nodes. This sets instance parameters and disables SYSDBA operating system authentication.

You need to run this command on all Oracle RAC nodes other than the node on which the Database Vault installation is performed. This step is required to enable the enhanced security features provided by Oracle Database Vault.

This section includes the following topics:

- Syntax for Using DVCA -action optionrac
- Procedure for Configuring Oracle Database Vault on RAC Nodes

Syntax for Using DVCA -action optionrac

The syntax for using `dvca -action optionrac` is as follows:

```
dvca -action optionrac
  -oh Oracle_home
  -instance Oracle_instance_name
  -dbname database_name
  -jdbc_str jdbc_connection_string
  -sys_passwd SYS_password
  [-silent]
  [-logfile ./dvca.log]
  [-nodecrypt]
  [-lockout]
```

In this specification:

- `-action` is the action to perform. optionrac performs the action of updating the instance parameters for the Oracle RAC instance. This flag also provides the option of disabling SYSDBA operating system access for the instance.

- `-racnode` is the host name of the Oracle RAC node on which the action is being performed. Do not include the domain name with the host name.
- `-oh` is the Oracle home for the Oracle RAC instance. Provide the `ORACLE_HOME` path.
- `-instance` is the name of the Database instance.
- `-dbname` is the database name.
- `-jdbc_str` is the JDBC connection string used to connect to the database. For example:

  ```
  jdbc:oracle:oci:@orcl1
  ```

- `-sys_password` is the password for the `SYS` user. If you use a cleartext password on the command line, you must include the `nodecrypt` option. If you omit the password, DVCA prompts you for it.
- `-logfile` is an optional flag to specify a log file name and location. You can enter an absolute path, or enter a path that is relative to the location of the *$ORACLE_HOME*/`bin` directory.
- `-silent` is the option to run in command line mode. This option is required if you are not running DVCA in an xterm window.
- `-nodecrypt` is the option to read plaintext passwords.
- `-lockout` is the flag to use to disable `SYSDBA` operating system authentication.

Procedure for Configuring Oracle Database Vault on RAC Nodes

To configure Oracle Database Vault on RAC nodes:

1. Ensure that the listener and database instance are running on the nodes on which you run DVCA.
2. At a command prompt, enter the DVCA command. For example:

   ```
   dvca -action optionrac
   -oh /path/to/oracle_home
   -jdbc_str jdbc:oracle:oci:@orcl1
   -racnode mynode
   -silent
   -logfile ./dvcalog.txt

   Enter SYS password: sys_password
   ```

Adding Languages to Oracle Database Vault

By default, Oracle Database Vault loads only the English language tables. You can use DVCA to add more languages to Oracle Database Vault by specifying the `addlanguages` flag to the `dvca -action` option.

This section includes the following topics:

- Syntax for Using DVCA -action addlanguages
- Procedure for Adding Languages to Oracle Database Vault

Syntax for Using DVCA -action addlanguages

The syntax for using `dvca -action addlanguages` is as follows:

```
dvca -action addlanguages
  -oh Oracle_home
  -instance Oracle_instance_name
  -dbname database_name
  -jdbc_str jdbc_connection_string
  -sys_passwd SYS_password
  -dvsys_passwd DVSYS_password
  -languages language_list
  [-owner_account DV_owner_account_name]
  [-owner_passwd DV_owner_account_password>]
  [-acctmgr_account DV_account_manager_account_name>]
  [-acctmgr_passwd DV_account_manager_password]
  [-silent]
  [-logfile ./dvca.log]
  [-nodecrypt]
  [-lockout]
  [-racnode node]
```

In this specification:

- `-action` is the action to perform. In this case the action is `addlanguages`.
- `-oh` is the Oracle home for the Oracle RAC instance. Provide the `ORACLE_HOME` path.
- `-instance` is the name of the Database instance.
- `-dbname` is the database name.
- `-sys_password` is the password for the `SYS` user. If you use a cleartext password on the command line, you must include the `nodecrypt` option. If you omit the password, DVCA prompts you for it.
- `-dvsys_password` is the password for the `DVSYS` user. If you use a cleartext password on the command line, you must include the `nodecrypt` option. If you omit the password, DVCA prompts you for it.
- `-jdbc_str` is the JDBC connection string used to connect to the database. For example:

  ```
  jdbc:oracle:oci:@orcl1
  ```

- `-languages` is the list of languages to be loaded. Provide the list of languages as a string in the following format.

  ```
  {"[en],[de],[es],[fr],[it],[ja],[ko],[pt_BR],[zh_CN],[zh_TW]"}
  ```

  Oracle Database Vault supports the following languages:

  `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

  For example, to load German and Spanish, you would enter the following:

  ```
   -languages {"de,es"}
  ```

- `-owner_account` is the Oracle Database Vault Owner account name.
- `-owner_passwd` is the Oracle Database Vault Owner account password. If you use a cleartext password on the command line, you must include the `nodecrypt` option. If you omit the password, DVCA prompts you for it.
- `-acctmgr_account` is the Oracle Database Vault Account Manager user name.
- `-acctmgr_passwd` is the Oracle Database Vault Account Manager password. If you use a cleartext password on the command line, you must include the `nodecrypt` option. If you omit the password, DVCA prompts you for it.
- `-logfile` is an optional flag to specify a log file name and location. You can enter an absolute path, or enter a path that is relative to the location of the *`$ORACLE_HOME`*`/bin` directory.
- `-silent` is the option to run in command line mode. This option is required if you are not running DVCA in an xterm window.
- `-nodecrypt` is the option to read plaintext passwords.
- `-lockout` is the flag to use to disable `SYSDBA` operating system authentication.

Procedure for Adding Languages to Oracle Database Vault

To add languages to Oracle Database Vault:

1. Disable Oracle Database Vault by completing the following steps under "Step 1: Disable Oracle Database Vault" on page B-2:
   - **Linux**: Steps 1, 2, and 3 (but not 4).
   - **Windows**: Steps 1, 2, 3, and 4 (but not 5).
2. Use DVCA to add the languages you want.

   For example:

   ```
   dvca -action addlanguages
     -oh /path/to/oracle_home
     -instance myinstance -dbname mydbname
     -jdbc_str jdbc:oracle:oci:@orcl1
     -languages {"es, ja"}
     -silent
     -logfile dvcalog.txt

   Enter SYS password: sys_password
   Enter DVSYS password: dvsys_password
   Enter owner password: owner_password
   Enter DV account manager password: dv_acct_password
   ```
3. Enable Oracle Database Vault by completing the following steps "Step 3: Enable Oracle Database Vault" on page B-6:
   - **Linux**: Steps 2, 3, 4, and 5 (but not 1).
   - **Windows**: Steps 2, 3, 4, and 5 (but not 1).

D

Oracle Database Vault Objects

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.

This chapter includes the following sections:

- Oracle Database Vault Schemas
- Oracle Database Vault Roles
- Oracle Database Vault Accounts
- Oracle Database Vault Public Views

Oracle Database Vault Schemas

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.

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.
- 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. SYSDBA 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 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 predefined administrative roles of the schema. "Oracle Database Vault Roles" on page D-2 describes Oracle Database Vault administrative roles in detail.
- Only the protected schema account DVSYS can grant predefined roles to users along with the ADMIN OPTION. This means that a grantee with the ADMIN OPTION can grant the role to another user without the ADMIN OPTION.
- 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.

Oracle Database Vault Roles

This section describes the default roles Oracle Database Vault provides. It includes the following topics:

- About Oracle Database Vault Roles
- Oracle Database Vault Owner Role, DV_OWNER
- Oracle Database Vault Realm DBA Role, DV_REALM_OWNER
- Oracle Database Vault Application Resource Owner Role, DV_REALM_RESOURCE
- Oracle Database Vault Configuration Administrator Role, DV_ADMIN
- Oracle Database Vault User Manager Role, DV_ACCTMGR
- Oracle Database Vault PUBLIC Role, DV_PUBLIC
- Oracle Database Vault Security Analyst Role, DV_SECANALYST

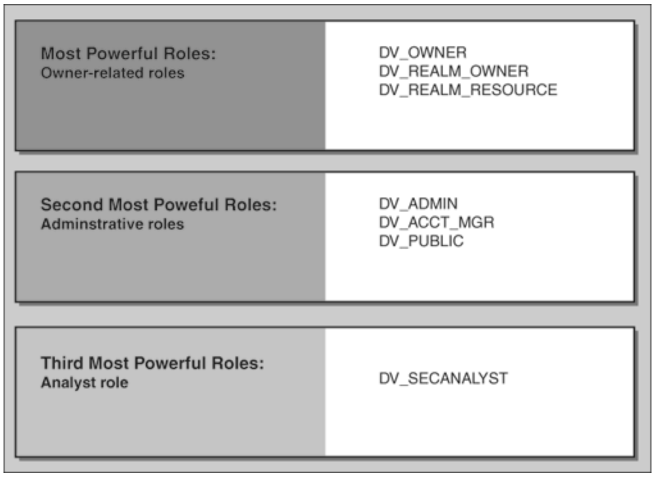
About Oracle Database Vault Roles

Oracle Database Vault provides a set of roles that are required for managing Oracle Database Vault. These roles are designed to implement the first level of separation of duties within the database, organized in the following hierarchy: The most powerful level is for the owner-related roles, DV_OWNER, DV_REALM_OWNER, and DV_REALM_RESOURCE. The next level beneath it is for the administrative roles, DV_ADMIN, DV_

ACCTMGR, and DV_PUBLIC. The third level is for the analyst-related role, DV_SECANALYST.

Figure D–1 illustrates the hierarchy of Oracle Database Vault roles, based on their levels of power.

Figure D–1 Hierarchy of Oracle Database Vault Roles



> **Note:** You can grant additional object privileges and roles to the Oracle Database Vault roles to extend their scope of privileges. For example, SYSDBA can grant object privileges to an Oracle Database Vault role as long as the object is not in the DVSYS schema or realm.

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

Table D–1 Privileges of Oracle Database Vault Roles

Privilege	DV_OWNER	DV_REALM_OWNER	DV_REALM_RESOURCE	DV_ADMIN	DV_ACCTMGR	DV_SECANALYST	DV_PUBLIC
DVSYS schema, EXECUTE	Yes	No	No	Yes	No	No	No
DVSYS packages, EXECUTE	Yes	No	No	Yes	No	No	No
DVSYS schema, SELECT	Yes	No	No	No	No	Yes[1]	Yes[2]
DVSYS schema, grant privileges on objects	No	No	No	No	No	No	Yes
DVF schema, EXECUTE	Yes	No	No	No	No	No	No
DVF schema, SELECT	No	No	No	No	No	Yes	No
Monitor Database Vault	Yes	No	No	Yes	No	Yes	Yes
Run Database Vault reports	Yes	No	No	Yes	No	Yes	Yes
SYS schema, SELECT	Yes	No	No	No	No	Yes, on some system views	No
SYSMAN schema, SELECT	No	No	No	No	No	Yes, portions of	No
CREATE, ALTER, DROP user accounts and profiles[3]	No	No	No	No	Yes	No	No
Manage objects in schemas that define a realm[4]	No	Yes	No	No	No	No	No

Table D–1 (Cont.) Privileges of Oracle Database Vault Roles

Privilege	DV_OWNER	DV_REALM_OWNER	DV_REALM_RESOURCE	DV_ADMIN	DV_ACCTMGR	DV_SECANALYST	DV_PUBLIC
RESOURCE role privileges[5]	No	No	Yes	No	No	No	No
CREATE SYNONYM	No	No	Yes	No	No	No	No
CREATE VIEW	No	No	Yes	No	No	No	No

[1] DV_SECANALYST can query DVSYS schema objects through Oracle Database Vault-supplied views only.

[2] DV_PUBLIC can query DVSYS schema objects through Oracle Database Vault-supplied views only.

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

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

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

Oracle Database Vault Owner Role, DV_OWNER

Use the DV_OWNER role manage the Oracle Database Vault roles and its configuration. The DV_OWNER role has the administrative capabilities that the DV_ADMIN role provides, and the reporting capabilities the DV_SECANALYST role provides. It 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 this guide, the example account that uses this role is MACSYS.) In addition to DV_ADMIN role, the DV_OWNER role has the GRANT ANY ROLE, ADMINISTER DATABASE TRIGGER, ALTER ANY TRIGGER privileges, and EXECUTE privileges on the SYS.DBMS_RLS package.

The first account, which is typically the account created during the installation as the Database Vault Owner, granted with this role and the ADMIN OPTION can grant any Oracle Database Vault roles (except DV_ACCTMGR) without the ADMIN OPTION to any account. Users granted this role also can run Oracle Database Vault reports and monitor Oracle Database Vault.

Anyone with the DV_OWNER role or privilege can grant the DV_OWNER role to another user. The account granted this role and with the ADMIN OPTION can revoke any granted protected schema 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 rights to grant or revoke the DV_OWNER role from any other database account.

The granting and revoking of all protected schema roles, including DV_OWNER, are enforced only by an instance with the Oracle executable linked with DV_ON, which enables Oracle Database Vault security. When the Oracle executable is linked with DV_OFF, then an instance can use an account GRANT ANY ROLE system privilege for GRANT and REVOKE operations.

Appendix B, "Enabling and Disabling Oracle Database Vault" shows how to use DV_ON and DV_OFF.

Oracle Database Vault Realm DBA Role, DV_REALM_OWNER

Use the DV_REALM_OWNER role to manage database objects in multiple schemas that define a realm. This role should be granted to the database account owner who would manage one or more schema database accounts within a realm and the roles associated with the realm. A user granted this role can use powerful system privileges like CREATE ANY, ALTER ANY, and DROP ANY within the realm.

The realm owner of the Oracle Data Dictionary realm, such as SYS, can grant this role to any given database account or role. Note that though this role has system privilege grants that SYS controls, it does not have the DV_OWNER or DV_ADMIN roles.

If you want to attach this role to a specific realm, you need to assign it to an account or business-related role, then authorize that account or role in the realm.

Oracle Database Vault Application Resource Owner Role, DV_REALM_RESOURCE

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.This role can be granted to a database account that will own database tables, objects, triggers, views, procedures, and so on that are used to support any database application. This is a role geared toward a schema type database account. The realm owner of the Oracle Data Dictionary realm, such as SYS, can grant this role to any database account or role. Note that though this role has system privilege grants that SYS controls, it does not have the DV_OWNER or DV_ADMIN privileges.

Oracle Database Vault Configuration Administrator Role, DV_ADMIN

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.

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 first user with the DV_ADMIN role and the ADMIN OPTION can grant this role without the ADMIN OPTION to any database account and revoke this role from another account.

The granting and revoking of protected schema roles, including DV_ADMIN, are enforced only by an instance with the Oracle executable linked with DV_ON, which enables Oracle Database Vault security features. When the Oracle executable is linked with DV_OFF, then an instance can use an account GRANT ANY ROLE system privilege for GRANT and REVOKE operations.

Appendix B, "Enabling and Disabling Oracle Database Vault" explains how to use DV_ON.

Oracle Database Vault User Manager Role, DV_ACCTMGR

Use the DV_ACCTMGR role to create and maintain database accounts and database profiles. A user who has been granted this role can use the CREATE, ALTER, and DROP statements for users or profiles. However, a person with this role cannot use the DROP or ALTER statements for the DVSYS account, nor change the DVSYS password.

> **Tip:** Oracle recommends that you add the user who has the DV_ACCTMGR role to the data dictionary realm so that this user can grant other users ANY privileges, if they need them. See "Step 1: Adding the SYSTEM User to the Data Dictionary Realm" on page 3-4 for instructions.

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 first account with the DV_ACCTMGR role and the ADMIN

OPTION can grant this role without the ADMIN OPTION to any given database account and revoke this role from another account.

The granting and revoking of protected schema roles are enforced only by an instance with the Oracle executable linked with DV_ON, which enables Oracle Database Vault. When the Oracle executable is linked with DV_OFF, then an instance can use an account with GRANT ANY ROLE system privilege for GRANT and REVOKE operations.

Appendix B, "Enabling and Disabling Oracle Database Vault" shows how to use DV_ON and DV_OFF.

Oracle Database Vault PUBLIC Role, DV_PUBLIC

Use the DV_PUBLIC role to grant privileges on specific objects in the DVSYS schema. Oracle Database Vault does not enable you to directly grant object privileges in the DVSYS schema to PUBLIC. You must grant an the object privilege on the DVSYS schema object the DV_PUBLIC role, and then grant DV_PUBLIC to PUBLIC. However, if you do this, it is important that you do not add more object privileges to the PUBLIC role. Doing so may undermine Oracle Database Vault security.

The following Oracle Database Vault objects are accessible through DV_PUBLIC:

- PL/SQL procedures and functions, described in "Oracle Database Vault Run-Time PL/SQL Procedures and Functions" on page H-1. These enable access control and Oracle Label Security processing in an Oracle database.
- PL/SQL factor functions, described in "Oracle Database Vault PL/SQL Factor Functions" on page H-5. For the DVF schema, these are functions for each factor defined. These are functions that you can use in rule sets to inspect the SQL statement that you want the rule set to protect.
- DVSYS.DBMS_MACSEC_ROLES package, described in Appendix F, "Oracle Database Vault DVSYS.DBMS_MACSEC_ROLES Package". This package enables you to check the authorization for a user or to set an Oracle Database Vault secure application role.
- DVSYS.DBMS_MACUTL package, described in Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package". This package is a set of general purpose utility functions that you can use throughout the application code you write for Oracle Database Vault.

Oracle Database Vault Security Analyst Role, DV_SECANALYST

Use the DV_SECANALYST role to run Oracle Database Vault reports and monitor Oracle Database Vault. (This role is also used for database-related reports.) In addition, this role enables you to check the DVSYS configuration by querying the DVSYS views described in "Oracle Database Vault Public Views" on page D-10. The DV_SECANALYST role has SELECT privileges on the DVSYS schema objects and portions of the SYS and SYSMAN schema objects for reporting on DVSYS- and DVF-related entities.

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 user with the DV_SECANALYST role and the ADMIN OPTION can grant this role without the ADMIN OPTION to any database account and revoke this role from another account.

The granting and revoking of protected schema roles are enforced only by an instance with the Oracle executable linked with DV_ON, which enables the Oracle Database Vault security features. When the Oracle executable is linked with DV_OFF, then an instance can use an account GRANT ANY ROLE system privilege for GRANT and REVOKE operations.

Appendix B, "Enabling and Disabling Oracle Database Vault" shows how to use DV_ON and DV_OFF.

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.

The Oracle Database Vault Account Manager account is granted the DV_ACCTMGR role. This account is used to manage database user accounts to facilitate separation of duties.

If you prefer to use an existing account to act as the Database Vault Owner and Account Manager, you can grant these roles to the account.

> **Note:** If you opt not to create the Oracle Database Vault Account Manager account during installation, then both the DV_OWNER and DV_ACCTMGR roles are granted to the Oracle Database Vault Owner user account.

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

Table D–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 an Oracle Database Vault Factor with an Oracle Label Security Policy" on page 9-4 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 D–3 lists some model database accounts that can act as a guide. (The accounts listed in Table D–3 serve as a guide to implementing Oracle Database Vault roles. These are not actual accounts that are created during installation.)

Table D–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 DVSYS realm. This account can: ■ Run DVSYS packages ■ Have EXECUTE privileges in the DVSYS schema ■ Grant privileges on the DVSYS schema objects ■ Select objects in the 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, or drop users ■ Create, alter, or drop profiles ■ Grant the DV_ACCTMGR role ■ Grant 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 ■ Have EXECUTE privileges in the DVSYS schema ■ 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 in the Oracle Database Vault Administration application.

Database Accounts Creation Scenarios

The general approach to creating database accounts and using the database roles provided in a database protected by Oracle Database Vault is as follows:

1. Log in as the Oracle Database Vault account manager (that is, using the DV_ACCTMGR role) to create the new database account.

2. Grant the new account the ability to create a database session.

3. Depending on the type of account being created, log in as the Oracle Data Dictionary realm owner or Oracle Database Vault realm owner to grant the appropriate roles required for the account.

4. Grant additional system or object privileges as required by the account.

The following examples demonstrate the uses of the Oracle Database Vault roles and database accounts. The examples assume the creation of an application schema type account named `bizapp`, a realm-owner type account named `mary`, an application end-user type account named `jiawen`, and a security administrator named `steve`.

These examples assume that you have added the `DV_ACCTMGR` role to the Data Dictionary realm. See "Step 1: Adding the SYSTEM User to the Data Dictionary Realm" on page 3-4 for instructions on how to do this.

Example D–1 Creating a Schema Account

```
SQL> CONNECT jgodfrey
-- connect using the DV_ACCTMGR role
Enter password: password
SQL> CREATE USER bizapp IDENTIFIED BY password;
-- provide session connectivity
SQL> GRANT CONNECT TO bizapp;


SQL> CONNECT SYS / AS SYSDBA
Enter password: password
-- provide the ability to create database objects
SQL> GRANT dv_realm_resource TO bizapp;
SQL> GRANT UNLIMITED TABLESPACE TO bizapp;


SQL> CONNECT bizapp
Enter password: password
SQL> CREATE TABLE bizapp.cases...;
```

Example D–2 Creating an Account for a Realm Owner

```
SQL> CONNECT jgodfrey
-- connect using the DV_ACCTMGR role
Enter password: password
SQL> CREATE USER mary IDENTIFIED BY password;
-- provide session connectivity
SQL> GRANT CONNECT TO mary;

SQL> CONNECT ebrown
-- connect using the DV_OWNER role
Enter password: password
-- provide ANY system privileges a realm owner would need
SQL> GRANT dv_realm_owner TO mary;

SQL> CONNECT mary
Enter password: password
SQL> ALTER TABLE bizapp.cases
```

Example D–3 Creating an Account for an Application User

```
SQL> CONNECT jgodfrey
-- connect using the DV_ACCTMGR role
Enter password: password
SQL> CREATE USER jiawen IDENTIFIED BY password;
SQL> DEFAULT TABLESPACE low_ts TEMPORARY TABLESPACE low_ts;
-- provide session connectivity
SQL> GRANT CONNECT TO jiawen;

-- the realm owner can manage privileges against realm objects
```

```
SQL> CONNECT mary
Enter password: password
SQL> GRANT SELECT ON bizapp.cases TO jiawen;

SQL> CONNECT jiawen
Enter password: password
-- query application tables
SQL> SELECT * FROM bizapp.cases
```

Example D–4 Creating an Account for a Security Administrator

```
SQL> CONNECT jgodfrey
-- connect using the DV_ACCTMGR role
Enter password: password
SQL> CREATE USER steve IDENTIFIED BY password;
SQL> DEFAULT TABLESPACE high_ts;
-- provide session connectivity
SQL> GRANT CONNECT TO steve;

-- allow execute privileges on DBMS_MACADM package
-- and the ability to query access control views
SQL> GRANT dv_admin TO steve


-- query and administer access control configuration
SQL> CONNECT steve
-- connect as the system administrator
Enter password: password
SQL> SELECT * FROM dvsys.dba_dv_factor;
SQL> EXEC dvsys.dbms_macadm.create_factor(...);
```

Oracle Database Vault Public Views

Oracle Database Vault provides a set of DBA-style views that can be accessed through the DV_SECANALYST role or the DV_ADMIN role. (Alternatively, you can run reports on Oracle Database Vault. See Chapter 11, "Oracle Database Vault Reports" for more information.) 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 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.

Table D–4 describes the Oracle Database Vault public views.

Table D–4 Oracle Database Vault Views

View	Description
DBA_DV_CODE	This view lists generic lookup codes for the user interface, error messages, constraint checking, and the like. These codes are used for the user interface, views, and for validating input in a translatable fashion. Each record contains a code group column that categorizes the code, a natural key, and an optional label and description. The following code groups are provided: ■ AUDIT_EVENTS: Contains the action numbers and action names that are used for the custom event audit trail records ■ BOOLEAN: A simple Yes/No or True/False lookup ■ DB_OBJECT_TYPE: The database object types that can be used for realm objects and command authorizations ■ DDL_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 ■ OPERATORS: The Boolean operators that can be used for identity maps ■ REALM_AUDIT: The options for auditing realm access or realm violations ■ RULESET_AUDIT: The options for auditing rule set execution or rule set errors ■ RULESET_EVALUATE: The options for determining the success or failure of a rule set based on all associated rules being true or any associated rule being true ■ RULESET_EVENT: The options to invoke a custom event handler when a rule set evaluates to Succeeds or Fails ■ RULESET_FAIL: The options to determine the run-time visibility of a rule set failing
DBA_DV_COMMAND_RULE	This view lists the SQL statement that are protected.
DBA_DV_FACTOR	This view lists the factors used.
DBA_DV_FACTOR_LINK	This view shows the relationships of each factor whose identity is determined by the association of child factors. The 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.

Table D–4 (Cont.) Oracle Database Vault Views

View	Description
DBA_DV_FACTOR_TYPE	This view lists the names and descriptions of factor types used in the system."
DBA_DV_IDENTITY	This view lists the identities for each factor.
DBA_DV_IDENTITY_MAP	This 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 will be joined by the OR operation, and for different factors, the maps will be 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).
DBA_DV_MAC_POLICY	This view lists the Oracle Label Security policies defined.
DBA_DV_MAC_POLICY_FACTOR	This 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.
DBA_DV_POLICY_LABEL	This view lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.
DBA_DV_PUB_PRIVS	This view lists the Oracle Database Vault privilege management reports used in the Oracle Database Vault Administrator (DV_ADMIN).
DBA_DV_REALM	This view lists the realms created.
DBA_DV_REALM_AUTH	This view lists the authorization of a named database user account or database role (GRANTEE) to access realm objects in a particular realm.
DBA_DV_REALM_OBJECT	This view lists the database schemas, or subsets of schemas with specific database objects contained therein, that are secured by the realms.
DBA_DV_ROLE	This view lists the Oracle Database secure application roles used in privilege management.
DBA_DV_RULE	This view lists the rules that have been defined. To find the rule sets that use specific rules, use the DBA_DV_RULE_SET_RULE view.
DBA_DV_RULE_SET	This view lists the rules sets that have been created.
DBA_DV_RULE_SET_RULE	This view lists rules that are associated with the rule sets used.
DBA_DV_USER_PRIVS	This view lists the privileges for a database account excluding privileges granted through PUBLIC.
DBA_DV_USER_PRIVS_ALL	This view lists the privileges for a database account including privileges granted through PUBLIC.

E

Oracle Database Vault DVSYS.DBMS_MACADM Package

The functions within the `DVSYS.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 `DVSYS.DBMS_MACADM` package is available only for users who have the `DV_ADMIN` or `DV_OWNER` role.

This section includes the following topics:

- Realm Functions Within DVSYS.DBMS_MACADM
- Rule Set Functions Within DVSYS.DBMS_MACADM
- Command Rule Functions Within DVSYS.DBMS_MACADM
- Factor Functions Within DVSYS.DBMS_MACADM
- Secure Application Role Functions Within DVSYS.DBMS_MACADM
- Oracle Label Security Policy Functions Within DVSYS.DBMS_MACADM

Realm Functions Within DVSYS.DBMS_MACADM

Table E–1 lists functions within the `DVSYS.DBMS_MACADM` package that you can use to configure realms. For constants that you can use with these functions, see Table G–1 on page G-1 for more information.

Chapter 4, "Configuring Realms" describes realms in detail. See also Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package" for a set of general purpose utility functions that you can use with the realm functions.

Table E–1 DVSYS.DBMS_MACADM Realm Configuration Functions

Function	Description
ADD_AUTH_TO_REALM Function	Authorizes a user or role to access a realm as a participant.
ADD_AUTH_TO_REALM Function	Authorizes a user or role to access a realm as an owner or participant (no rule set).
ADD_AUTH_TO_REALM Function	Authorizes a user or role to access a realm as a participant. Optionally, you can specify a rule set for the authorization.
ADD_AUTH_TO_REALM Function	Authorizes a user or role to access a realm as a participant or owner. Optionally, you can specify a rule set for the authorization.
ADD_OBJECT_TO_REALM Function	Registers a set of objects for realm protection.

Table E–1 (Cont.) DVSYS.DBMS_MACADM Realm Configuration Functions

Function	Description
CREATE_REALM Function	Creates a realm.
DELETE_AUTH_FROM_REALM Function	Removes the authorization of a user or role to access a realm.
DELETE_OBJECT_FROM_REALM Function	Removes a set of objects from realm protection.
DELETE_REALM Function	Deletes a realm.
DELETE_REALM_CASCADE Function	Deletes a realm, including its related Database Vault configuration information.
RENAME_REALM Function	Renames a realm. The name change takes effect everywhere the realm is used.
UPDATE_REALM Function	Updates a realm.
UPDATE_REALM_AUTH Function	Updates the authorization of a user or role to access a realm.

ADD_AUTH_TO_REALM Function

This function authorizes a user or role to access a realm as a participant. The person running this function cannot add himself or herself to the realm as a realm participant.

Syntax

```
ADD_AUTH_TO_REALM(
  realm_name  VARCHAR2,
  grantee     VARCHAR2);
```

Parameters

Table E–2 ADD_AUTH_TO_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`grantee`	User or role name to authorize as a participant. To find the existing users and roles in the current database instance, use the `DBA_USERS` and `DBA_ROLES` views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, use the `DVA_DV_REALM_AUTH` view. To find existing secure application roles used in privilege management, use the `DBA_DV_ROLE` view. Both are described in "Oracle Database Vault Public Views" on page D-10.

ADD_AUTH_TO_REALM Function

This function authorizes a user or role to access a realm as an owner or a participant. The person running this function cannot add himself or herself to the realm as a realm owner or participant.

Syntax

```
ADD_AUTH_TO_REALM(
  realm_name   VARCHAR2,
  grantee      VARCHAR2,
  auth_options NUMBER);
```

Parameters

Table E–3 ADD_AUTH_TO_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`grantee`	User or role name to authorize as owner or participant. To find the existing users and roles in the current database instance, use the `DBA_USERS` and `DBA_ROLES` views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, use the `DVA_DV_REALM_AUTH` view. To find existing secure application roles used in privilege management, use the `DBA_DV_ROLE` view. Both are described in "Oracle Database Vault Public Views" on page D-10.
`auth_options`	Specify one of the following ways to authorize the realm: ■ `0`: Participant. ■ `1`: Owner See "Defining Realm Authorization" on page 4-5 for more information on participants and owners.

ADD_AUTH_TO_REALM Function

This function authorizes a user or role to access a realm as a participant. The person running this function cannot add himself or herself to the realm as a realm participant. Optionally, you can specify a rule set to check before allowing the authorization to proceed.

Syntax

```
ADD_AUTH_TO_REALM(
  realm_name    VARCHAR2,
  grantee       VARCHAR2,
  rule_set_name VARCHAR2);
```

Parameters

Table E–4 ADD_AUTH_TO_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`grantee`	User or role name to authorize as participant. To find the existing users and roles in the current database instance, use the `DBA_USERS` and `DBA_ROLES` views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, use the `DVA_DV_REALM_AUTH` view. To find existing secure application roles used in privilege management, use the `DBA_DV_ROLE` view. Both are described in "Oracle Database Vault Public Views" on page D-10.

Table E–4 (Cont.) ADD_AUTH_TO_REALM Parameters

Parameter	Description
`rule_set_name`	Rule set to check before authorizing (optional). If the rule set evaluates to TRUE, then the authorization is allowed. To find the available rule sets, use the `DBA_DV_RULE_SET` view. To find rules that are associated with the rule sets, use the `DBA_DB_RULE_SET_RULE` view. Both are described in "Oracle Database Vault Public Views" on page D-10.

ADD_AUTH_TO_REALM Function

This function authorizes a user or role to access a realm as a participant or owner. The person running this function cannot add himself or herself to the realm as a realm owner or participant. Optionally, you can specify a rule set to check before authorizing.

Syntax

```
ADD_AUTH_TO_REALM(
  realm_name    VARCHAR2,
  grantee       VARCHAR2,
  rule_set_name VARCHAR2,
  auth_options  NUMBER);
```

Parameters

Table E–5 ADD_AUTH_TO_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`grantee`	User or role name to authorize as owner or participant. To find the available users and roles, use the `DBA_USERS` and `DBA_ROLES` views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, use the `DVA_DV_REALM_AUTH` view, described in "Oracle Database Vault Public Views" on page D-10.
`rule_set_name`	Rule set to check before authorizing (optional). If the rule set evaluates to TRUE, then the authorization is allowed. To find the available rule sets, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.
`auth_options`	Specify one of the following ways to authorize the realm: ■ `0`: Participant. ■ `1`: Owner See "Defining Realm Authorization" on page 4-5 for more information on participants and owners.

ADD_OBJECT_TO_REALM Function

This function registers a set of objects for realm protection.

Syntax

```
ADD_OBJECT_TO_REALM(
  realm_name   VARCHAR2,
  object_owner VARCHAR2,
  object_name  VARCHAR2,
  object_type  VARCHAR2);
```

Parameters

Table E–6 ADD_OBJECT_TO_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`object_owner`	Object owner to own this realm. To find the available users, use the `DBA_USERS` view, described in *Oracle Database Reference*. To find the authorization of a particular user, use the `DVA_DV_REALM_AUTH` view, described in "Oracle Database Vault Public Views" on page D-10.
`object_name`	Object name. (The wildcard % is allowed. See "Object Name" under "Creating Realm-Secured Objects" on page 4-4 for exceptions to the wildcard %.) To find the available objects, use the `ALL_OBJECTS` view, described in *Oracle Database Reference*. To find objects that are secured by existing realms, use the `DBA_DV_REALM_OBJECT` view, described in "Oracle Database Vault Public Views" on page D-10.
`object_type`	Object type, such as `TABLE`, `INDEX`, or `ROLE`. (The wildcard % is allowed. See "Object Types" under "Creating Realm-Secured Objects" on page 4-4 for exceptions to the wildcard %.)

CREATE_REALM Function

This function creates a realm. After you create the realm, use the following functions to complete the realm definition:

- `ADD_OBJECT_TO_REALM` function registers one or more objects for the realm.
- `ADD_AUTH_TO_REALM` functions authorize users or roles for the realm.

Syntax

```
CREATE_REALM(
  realm_name    VARCHAR2,
  description   VARCHAR2,
  enabled       VARCHAR2,
  audit_options NUMBER);
```

Parameters

Table E–7 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, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`description`	Description of the purpose of the realm, up to 1024 characters in mixed-case.
`enabled`	`Y` (yes) enables realm checking; `N` (no) disables it. The default is `Y`.
`audit_options`	Specify one of the following ways to audit the realm: ■ `0`: Disables 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.

DELETE_AUTH_FROM_REALM Function

This function removes the authorization of a user or role to access a realm.

Syntax

```
DELETE_AUTH_FROM_REALM(
  realm_name VARCHAR2,
  grantee    VARCHAR2);
```

Parameters

Table E–8 DELETE_AUTH_FROM_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`grantee`	User or role name. To find the authorization of a particular user or role, use the `DVA_DV_REALM_AUTH` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_OBJECT_FROM_REALM Function

This function removes a set of objects from realm protection.

Syntax

```
DELETE_OBJECT_FROM_REALM(
  realm_name   VARCHAR2,
  object_owner VARCHAR2,
  object_name  VARCHAR2,
  object_type VARCHAR2);
```

Parameters

Table E–9 DELETE_OBJECT_FROM_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`object_owner`	Database schema owner. To find the available users, use the `DBA_USERS` view, described in *Oracle Database Reference*. To find the authorization of a particular user, use the `DVA_DV_REALM_AUTH` view, described in "Oracle Database Vault Public Views" on page D-10.
`object_name`	Object name. (The wildcard % is allowed. See "Object Name" under "Creating Realm-Secured Objects" on page 4-4 for exceptions to the wildcard %.) To find objects that are secured by existing realms, use the `DBA_DV_REALM_OBJECT` view, described in "Oracle Database Vault Public Views" on page D-10.
`object_type`	Object type, such as `TABLE`, `INDEX`, or `ROLE`. (The wildcard % is allowed. See "Object Types" under "Creating Realm-Secured Objects" on page 4-4 for exceptions to the wildcard %.)

DELETE_REALM Function

This function deletes a realm but does not remove its associated objects and authorizations. Before you delete a realm, you can locate its associated objects by querying the `DBA_DV_REALM_OBJECT` view, described in"Oracle Database Vault Public Views" on page D-10.

If you want to remove the associated objects and authorizations as well as the realm, see "DELETE_REALM_CASCADE Function" on page E-7.

Syntax

```
DELETE_REALM(
  realm_name VARCHAR2);
```

Parameters

Table E–10 DELETE_REALM Parameter

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_REALM_CASCADE Function

This function deletes a realm, including its related Database Vault configuration information that specifies who is authorized (`dba_dv_realm_auth`) and what objects are protected (`dba_dv_realm_object`). It does not delete the actual database objects or users. To find a listing of the realm-related objects, use the DBA_DV_REALM view. To find its authorizations, query DBA_DV_REALM_AUTH. Both are described under "Oracle Database Vault Public Views" on page D-10.

Syntax

```
DELETE_REALM_CASCADE(
  realm_name VARCHAR2);
```

Parameters

Table E–11 DELETE_REALM_CASCADE Parameter

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, use the DBA_DV_REALMS view, described in "Oracle Database Vault Public Views" on page D-10.

RENAME_REALM Function

This function renames a realm. The name change takes effect everywhere the realm is used.

Syntax

```
RENAME_REALM(
  realm_name VARCHAR2,
  new_name   VARCHAR2);
```

Parameters

Table E–12 RENAME_REALM Parameters

Parameter	Description
realm_name	Current realm name. To find the existing realms in the current database instance, use the DBA_DV_REALMS view, described in "Oracle Database Vault Public Views" on page D-10.
new_name	New realm name, up to 90 characters in mixed-case.

UPDATE_REALM Function

This function updates a realm.

Syntax

```
UPDATE_REALM(
  realm_name    VARCHAR2,
  description   VARCHAR2,
  enabled       VARCHAR2,
  audit_options NUMBER);
```

Parameters

Table E–13 UPDATE_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`description`	Description of the purpose of the realm, up to 1024 characters in mixed-case.
`enabled`	`Y` (yes) enables realm checking; `N` (no) disables it. The default is `Y`.
`audit_options`	Specify one of the following ways to audit the realm: ■ `0`: Disables 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.

UPDATE_REALM_AUTH Function

Updates the authorization of a user or role to access a realm.

Syntax

```
UPDATE_REALM_AUTH(
  realm_name    VARCHAR2,
  grantee       VARCHAR2,
  rule_set_name VARCHAR2,
  auth_options  NUMBER);
```

Parameters

Table E–14 UPDATE_REALM_AUTH Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, use the `DBA_DV_REALMS` view, described in "Oracle Database Vault Public Views" on page D-10.
`grantee`	User or role name. To find the available users and roles, use the `DBA_USERS` and `DBA_ROLES` views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, use the `DVA_DV_REALM_AUTH` view. To find existing secure application roles used in privilege management, use the `DBA_DV_ROLE` view. Both are described in "Oracle Database Vault Public Views" on page D-10.

Table E–14 (Cont.) UPDATE_REALM_AUTH Parameters

Parameter	Description
`rule_set_name`	Rule set to check before authorizing (optional). If the rule set evaluates to TRUE, then the authorization is allowed. To find the available rule sets, use the `DBA_DV_RULE_SET` view. To find rules that are associated with the rule sets, use the `DBA_DB_RULE_SET_RULE` view. Both are described in "Oracle Database Vault Public Views" on page D-10.
`auth_options`	Specify one of the following ways to authorize the realm: ■ `0`: 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. ■ `1`: Owner. This account or role has the same privileges as the realm participant, plus the authorization to grant or revoke realm-secured database roles. A realm can have more than one owner.

Rule Set Functions Within DVSYS.DBMS_MACADM

Table E–15 lists functions within the `DVSYS.DBMS_MACADM` package that you can use to configure rule sets.

Chapter 5, "Configuring Rule Sets" describes rule sets in detail. See also Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility functions that you can use with the rule set functions.

Table E–15 DVSYS.DBMS_MACADM Rule Set Configuration Functions

Function	Description
ADD_RULE_TO_RULE_SET Function	Adds an enabled or disabled rule to the end of a rule set.
ADD_RULE_TO_RULE_SET Function	Adds a rule to a rule set and lets you specify its order within the rule set.
ADD_RULE_TO_RULE_SET Function	Adds a rule to a rule set.
CREATE_RULE Function	Creates a rule.
CREATE_RULE_SET Function	Creates a rule set.
DELETE_RULE Function	Deletes a rule.
DELETE_RULE_FROM_RULE_SET Function	Deletes a rule from a rule set.
DELETE_RULE_SET Function	Deletes a rule set.
RENAME_RULE Function	Renames a rule. The name change takes effect everywhere the rule is used.
RENAME_RULE_SET Function	Renames a rule set. The name change takes effect everywhere the rule set is used.
SYNC_RULES Function	Synchronizes the rules in Oracle Database Vault and Advanced Queuing Rules engine. You must perform this operation immediately after a rollback of an **Add**, **Delete**, or **Modify** rule operation.
UPDATE_RULE Function	Updates a rule.
UPDATE_RULE_SET Function	Updates a rule set.

ADD_RULE_TO_RULE_SET Function

This function adds an enabled or disabled rule to a rule set, and lets you specify its order within the rule set.

Syntax

```
ADD_RULE_TO_RULE_SET(
  rule_set_name  VARCHAR2,
  rule_name      VARCHAR2,
  rule_order     NUMBER,
  enabled        VARCHAR2);
```

Parameters

Table E–16 ADD_RULE_TO_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.
`rule_name`	Rule to add to the rule set. To find existing rules, use the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, use `DBA_DV_RULE_SET_RULE`. Both are described in "Oracle Database Vault Public Views" on page D-10.
`rule_order`	Does not apply to this release. The order in which rules appear affects performance. See "Improving Performance by Setting the Order in Which Rules Appear in a Rule Set" on page 5-8 for more information.
`enabled`	`Y` (yes) enables rule checking; `N` (no) disables it. The default is `Y`. You can also enter the following: ▪ `DBMS_MACUTIL.G_YES` ▪ `DBMS_MACUTIL.G_NO`

ADD_RULE_TO_RULE_SET Function

This function adds a rule to a rule set and lets you specify its order within the rule set.

Syntax

```
ADD_RULE_TO_RULE_SET(
  rule_set_name  VARCHAR2,
  rule_name      VARCHAR2,
  rule_order     NUMBER);
```

Parameters

Table E–17 ADD_RULE_TO_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.

Table E–17 (Cont.) ADD_RULE_TO_RULE_SET Parameters

Parameter	Description
`rule_name`	Rule to add to the rule set. To find existing rules, use the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, use `DBA_DV_RULE_SET_RULE`. Both are described in "Oracle Database Vault Public Views" on page D-10.
`rule_order`	Does not apply to this release. The order in which rules appear affects performance. See "Improving Performance by Setting the Order in Which Rules Appear in a Rule Set" on page 5-8 for more information.

ADD_RULE_TO_RULE_SET Function

This function adds a rule to a rule set.

Syntax

```
ADD_RULE_TO_RULE_SET(
  rule_set_name VARCHAR2,
  rule_name     VARCHAR2);
```

Parameters

Table E–18 ADD_RULE_TO_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.
`rule_name`	Rule to add to the rule set. To find existing rules in the current database instance, use the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, use `DBA_DV_RULE_SET_RULE`. Both are described in "Oracle Database Vault Public Views" on page D-10.

CREATE_RULE Function

This function creates a rule.

Syntax

```
CREATE_RULE(
  rule_name  VARCHAR2,
  rule_expr  VARCHAR2);
```

Parameters

Table E–19 CREATE_RULE Parameters

Parameter	Description
`rule_name`	Rule name, up to 90 characters in mixed-case. To find existing rules in the current database instance, use the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, use `DBA_DV_RULE_SET_RULE`. Both are described in "Oracle Database Vault Public Views" on page D-10.

Table E–19 (Cont.) CREATE_RULE Parameters

Parameter	Description
`rule_expr`	PL/SQL `BOOLEAN` expression. See "Creating a New Rule" on page 5-5 for more information on rule expressions.

CREATE_RULE_SET Function

This function creates a rule set. After you create a rule set, you can use the `CREATE_RULE` and `ADD_RULE_TO_RULE` set functions to create and add rules to the rule set.

Syntax

```
CREATE_RULE_SET(
  rule_set_name    VARCHAR2,
  description      VARCHAR2,
  enabled          VARCHAR2,
  eval_options     NUMBER,
  audit_options    NUMBER,
  fail_options     NUMBER,
  fail_message     VARCHAR2,
  fail_code        NUMBER,
  handler_options  NUMBER,
  handler          VARCHAR2);
```

Parameters

Table E–20 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, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.
`description`	Description of the purpose of the rule set, up to 1024 characters in mixed-case.
`enabled`	`Y` (yes) enables rule set checking; `N` (no) disables it. The default is `Y`.
`eval_options`	If you plan to assign more than one rule to the rule set, enter one of the following settings: ■ `1`: All rules in the rule set must evaluate to true for the rule set itself to evaluate to true. ■ `2`: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to true.
`audit_options`	Select one of the following settings: ■ `0`: Disables auditing ■ `POWER(2,0)`: Audits if the rule set evaluates to false (fails). ■ `POWER(2,1)`: Audits whenever the rule set is used. See "Audit Options" on page 5-3 for more information.

Table E–20 (Cont.) CREATE_RULE_SET Parameters

Parameter	Description
`fail_options`	Options for reporting factor errors: - `1`: Shows an error message. - `2`: Does not show an error message. See "Error Handling Options" on page 5-4 for more information.
`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 negative number in the range of -20000 to -20999, to associate with the `fail_message`.
`handler_options`	Select one of the following settings: - `0`: Disables error handling. - `POWER(2,0)`: Call handler on rule set failure. - `POWER(2,1)`: Call handler on rule set success. See "Error Handling Options" on page 5-4 for more information.
`handler`	Custom event handler logic. See "Error Handling Options" on page 5-4 for more information.

DELETE_RULE Function

This function deletes a rule.

Syntax

```
DELETE_RULE(
  rule_name VARCHAR2);
```

Parameter

Table E–21 DELETE_RULE Parameter

Parameter	Description
`rule_name`	Rule set name. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_RULE_FROM_RULE_SET Function

This function deletes a rule from a rule set.

Syntax

```
DELETE_RULE_FROM_RULE_SET(
  rule_set_name VARCHAR2,
  rule_name     VARCHAR2);
```

Parameters

Table E–22 DELETE_RULE_FROM_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.
`rule_name`	Rule to remove from the rule set. To find rules that have been associated with rule sets, use `DBA_DV_RULE_SET_RULE`. Both are described in "Oracle Database Vault Public Views" on page D-10.

DELETE_RULE_SET Function

This function deletes a rule set.

Syntax

```
DELETE_RULE_SET(
  rule_set_name VARCHAR2);
```

Parameters

Table E–23 DELETE_RULE_SET Parameter

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.

RENAME_RULE Function

This function renames a rule. The name change takes effect everywhere the rule is used.

Syntax

```
RENAME_RULE(
  rule_name  VARCHAR2,
  new_name   VARCHAR2);
```

Parameters

Table E–24 RENAME_RULE Parameters

Parameter	Description
`rule_name`	Rule name. To find existing rules in the current database instance, use the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, use `DBA_DV_RULE_SET_RULE`. Both are described in "Oracle Database Vault Public Views" on page D-10.
`new_name`	New rule name, up to 90 characters in mixed-case.

RENAME_RULE_SET Function

This function renames a rule set. The name change takes effect everywhere the rule set is used.

Syntax

```
RENAME_RULE_SET(
  rule_set_name VARCHAR2,
  new_name      VARCHAR2);
```

Parameters

Table E–25 RENAME_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Current rule set name.
`new_name`	New rule set name, up to 90 characters in mixed-case. Spaces are allowed.

SYNC_RULES Function

This function synchronizes the rules in Oracle Database Vault and Advanced Queuing Rules engine. You must perform this operation immediately after a rollback of an **Add**, **Delete**, or **Modify** rule operation.

Syntax

```
SYNC_RULES();
```

Parameters

None.

UPDATE_RULE Function

This function updates a rule.

Syntax

```
UPDATE_RULE(
  rule_name  VARCHAR2,
  rule_expr  VARCHAR2);
```

Parameters

Table E–26 UPDATE_RULE Parameters

Parameter	Description
`rule_name`	Rule name. To find existing rules in the current database instance, use the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, use `DBA_DV_RULE_SET_RULE`. Both are described in "Oracle Database Vault Public Views" on page D-10.
`rule_expr`	PL/SQL `BOOLEAN` expression. See "Creating a New Rule" on page 5-5 for more information on rule expressions.

UPDATE_RULE_SET Function

This function updates a rule set.

Syntax

```
UPDATE_RULE_SET(
  rule_set_name    VARCHAR2,
  description      VARCHAR2,
  enabled          VARCHAR2,
  eval_options     NUMBER,
  audit_options    NUMBER,
  fail_options     NUMBER,
  fail_message     VARCHAR2,
  fail_code        NUMBER,
  handler_options  NUMBER,
  handler          VARCHAR2);
```

Parameters

Table E–27 UPDATE_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.
`description`	Description of the purpose of the rule set, up to 1024 characters in mixed-case.
`enabled`	`Y` (yes) enables rule set checking; `N` (no) disables it. The default is `Y`.
`eval_options`	If you plan to assign more than one rule to the rule set, enter one of the following settings: ■ `1`: All rules in the rule set must evaluate to true for the rule set itself to evaluate to true. ■ `2`: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to true.
`audit_options`	Select one of the following settings: ■ `0`: Disables auditing ■ `POWER(2,0)`: Audits if the rule set evaluates to false (fails). ■ `POWER(2,1)`: Audits whenever the rule set is used. See "Audit Options" on page 5-3 for more information.
`fail_options`	Options for reporting factor errors: ■ `1`: Shows an error message. ■ `2`: Does not show an error message. See "Error Handling Options" on page 5-4 for more information.
`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 negative number in the range of -20000 to -20999, to associate with the `fail_message`.

Table E–27 (Cont.) UPDATE_RULE_SET Parameters

Parameter	Description
`handler_options`	Select one of the following settings: ■ `0`: Disables error handling. ■ `POWER(2,0)`: Call handler on rule set failure. ■ `POWER(2,1)`: Call handler on rule set success. See "Error Handling Options" on page 5-4 for more information.
`handler`	Custom event handler logic. See "Error Handling Options" on page 5-4 for more information.

Command Rule Functions Within DVSYS.DBMS_MACADM

Table E–28 lists functions within the `DVSYS.DBMS_MACADM` package that you can use to configure command rules.

Chapter 6, "Configuring Command Rules" describes command rules in detail. See also Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility functions that you can use with the command rule functions.

Table E–28 DVSYS.DBMS_MACADM Command Rule Configuration Functions

Function	Description
CREATE_COMMAND_RULE Function	Creates a command rule and associates it with a rule set.
DELETE_COMMAND_RULE Function	Drops a command rule declaration.
UPDATE_COMMAND_RULE Function	Updates a command rule declaration.

CREATE_COMMAND_RULE Function

This function creates a command rule and associates it with a rule set.

Syntax

```
CREATE_COMMAND_RULE(
  command        VARCHAR2,
  rule_set_name  VARCHAR2,
  object_owner   VARCHAR2,
  object_name    VARCHAR2,
  enabled        VARCHAR2);
```

Parameters

Table E–29 CREATE_COMMAND_RULE Parameters

Parameter	Description
`command`	SQL statement to protect. See *Oracle Database SQL Language Reference* for more information of 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, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.

Table E–29 (Cont.) CREATE_COMMAND_RULE Parameters

Parameter	Description
`object_owner`	Database schema owner for this command rule. To find the available users, use the `DBA_USERS` view, described in *Oracle Database Reference*. See also "Object Owner" in "Creating and Editing a Command Rule" on page 6-4 for more information about command rule owners.
`object_name`	Object name. (The wildcard % is allowed. See "Object Name" in "Creating and Editing a Command Rule" on page 6-4 for more information about objects protected by command rules.) To find the available objects, use the `ALL_OBJECTS` view, described in *Oracle Database Reference*.
`enabled`	`Y` (yes) enables command rule checking; `N` (no) disables it. The default is `Y`.

DELETE_COMMAND_RULE Function

This function drops a command rule declaration.

Syntax

```
DELETE_COMMAND_RULE(
  command      VARCHAR2,
  object_owner VARCHAR2,
  object_name  VARCHAR2);
```

Parameters

Table E–30 DELETE_COMMAND_RULE Parameters

Parameter	Description
`command`	SQL statement the command rule protects. See *Oracle Database SQL Language Reference* for more information of SQL statements.
`object_owner`	Database schema owner for this command rule. To find the available users in the current database instance, use the `DBA_USERS` view, described in *Oracle Database Reference*. See also "Object Owner" in "Creating and Editing a Command Rule" on page 6-4 for more information about command rule owners.
`object_name`	Object name. (The wildcard % is allowed. See "Object Name" in "Creating and Editing a Command Rule" on page 6-4 for more information about objects protected by command rules.) To find the available objects, use the `ALL_OBJECTS` view, described in *Oracle Database Reference*.

UPDATE_COMMAND_RULE Function

This function updates a command rule declaration.

Syntax

```
UPDATE_COMMAND_RULE(
  command        VARCHAR2,
  rule_set_name  VARCHAR2,
  object_owner   VARCHAR2,
  object_name    VARCHAR2,
  enabled        VARCHAR2);
```

Parameters

Table E–31 UPDATE_COMMAND_RULE Parameters

Parameter	Description
`command`	SQL statement to protect. See *Oracle Database SQL Language Reference* for more information of 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, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.
`object_owner`	Database schema owner for this command rule. To find the available users, use the `DBA_USERS` view, described in *Oracle Database Reference*. See also "Object Owner" in "Creating and Editing a Command Rule" on page 6-4 for more information about command rule owners.
`object_name`	Object name. (The wildcard % is allowed. See "Object Name" in "Creating and Editing a Command Rule" on page 6-4 for more information about objects protected by command rules.) To find the available objects, use the `ALL_OBJECTS` view, described in *Oracle Database Reference*.
`enabled`	`Y` (yes) enables command rule checking; `N` (no) disables it. The default is `Y`.

Factor Functions Within DVSYS.DBMS_MACADM

Table E–32 lists functions within the `DVSYS.DBMS_MACADM` package that you can use to configure factors.

Chapter 7, "Configuring Factors" describes factors in detail. See also Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility functions that you can use with the factor functions.

Table E–32 DVSYS.DBMS_MACADM Factor Configuration Functions

Function	Description
ADD_FACTOR_LINK Function	Specifies a parent-child relationship for two factors.
ADD_POLICY_FACTOR Function	Specifies that the label for a factor contributes to the Oracle Label Security label for a policy.
CHANGE_IDENTITY_FACTOR Function	Associates an identity with a different factor.
CHANGE_IDENTITY_VALUE Function	Updates the value of an identity.
CREATE_DOMAIN_IDENTITY Function	Adds an Oracle Real Application Clusters (RAC) database node to the domain factor identities and labels it according to the Oracle Label Security policy.
CREATE_FACTOR Function	Creates a factor.
CREATE_FACTOR_TYPE Function	Creates a factor type.
CREATE_IDENTITY Function	Creates an identity.
CREATE_IDENTITY_MAP Function	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 Function	Deletes a factor.

Table E–32 (Cont.) DVSYS.DBMS_MACADM Factor Configuration Functions

Function	Description
DELETE_FACTOR_LINK Function	Removes a parent-child relationship for two factors.
DELETE_FACTOR_TYPE Function	Deletes a factor type.
DELETE_IDENTITY Function	Removes an identity.
DELETE_IDENTITY_MAP Function	Removes an identity map from a factor.
DROP_DOMAIN_IDENTITY Function	Removes an Oracle Real Application Clusters (RAC) database node from a domain.
GET_INSTANCE_INFO Function	Returns information from the `SYS.V_$INSTANCE` view; returns a `VARCHAR2` value.
GET_SESSION_INFO Function	Returns information from the `SYS.V_$SESSION` view for the current session; returns a `VARCHAR2` value.
RENAME_FACTOR Function	Renames a factor. The name change takes effect everywhere the factor is used.
RENAME_FACTOR_TYPE Function	Renames a factor type. The name change takes effect everywhere the factor type is used.
UPDATE_FACTOR Function	Updates a factor.
UPDATE_FACTOR_TYPE Function	Updates the description of a factor type.
UPDATE_IDENTITY Function	Updates the trust_level of a factor identity.

ADD_FACTOR_LINK Function

This function specifies a parent-child relationship for two factors.

Syntax

```
ADD_FACTOR_LINK(
  parent_factor_name VARCHAR2,
  child_factor_name  VARCHAR2,
  label_indicator    VARCHAR2);
```

Parameters

Table E–33 ADD_FACTOR_LINK Parameters

Parameter	Description
`parent_factor_name`	Parent factor name. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`child_factor_name`	Child factor name. To find the relationships of existing factors whose identities are determined by the association of child factors, use the `DBA_DV_FACTOR_LINK` view, described in "Oracle Database Vault Public Views" on page D-10.

Table E–33 (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 `Y` (for yes) or `N` (for no). You can also use the following constants: ■ `DBMS_MACUTIL.G_YES` ■ `DBMS_MACUTIL.G_NO` To find the Oracle Label Security policies and labels associated with factors, use the following views, described in "Oracle Database Vault Public Views" on page D-10: ■ `DBA_DV_MAC_POLICY`: Lists Oracle Label Security policies defined in the current database instance. ■ `DBA_DV_MAC_POLICY_FACTOR`: Lists the factors that are associated with Oracle Label Security policies for the current database instance. ■ `DBA_DV_POLICY_LABEL`: Lists the Oracle Label Security label for each factor identifier in the `DBA_DV_IDENTITY` view for each policy.

ADD_POLICY_FACTOR Function

This function specifies that the label for a factor contributes to the Oracle Label Security label for a policy.

Syntax

```
ADD_POLICY_FACTOR(
  policy_name  VARCHAR2,
  factor_name  VARCHAR2);
```

Parameters

Table E–34 ADD_POLICY_FACTOR Parameters

Parameter	Description
`policy_name`	Oracle Label Security policy name. To find the policies defined in the current database instance, use the `DBA_DV_MAC_POLICY` view. To find factors that are associated with Oracle Label Security policies, use `DBA_DV_MAC_POLICY_FACTOR`. Both are described in "Oracle Database Vault Public Views" on page D-10.
`factor_name`	Factor name. To find existing factors, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.

CHANGE_IDENTITY_FACTOR Function

This function associates an identity with a different factor.

Syntax

```
CHANGE_IDENTITY_FACTOR(
  factor_name     VARCHAR2,
  value           VARCHAR2,
  new_factor_name VARCHAR2);
```

Parameters

Table E–35 CHANGE_IDENTITY_FACTOR Parameters

Parameter	Description
factor_name	Current factor name. To find existing factors in the current database instance, use the DBA_DV_FACTOR view, described in "Oracle Database Vault Public Views" on page D-10.
value	Value of the identity to update. To find existing identities for each factor in the current database instance, use the DBA_DV_IDENTITY view. To find current identity mappings, use the DBA_DV_IDENTITY_MAP view. Both are described in "Oracle Database Vault Public Views" on page D-10.
new_factor_name	Name of the factor to associate with the identity.

CHANGE_IDENTITY_VALUE Function

This function updates the value of an identity.

Syntax

```
CHANGE_IDENTITY_VALUE(
  factor_name  VARCHAR2,
  value        VARCHAR2,
  new_value    VARCHAR2);
```

Parameters

Table E–36 CHANGE_IDENTITY_VALUE Parameters

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, use the DBA_DV_FACTOR view, described in "Oracle Database Vault Public Views" on page D-10.
value	Current value associated with the identity. To find existing identities for each factor in the current database instance, use the DBA_DV_IDENTITY view. To find current identity mappings, use the DBA_DV_IDENTITY_MAP view. Both are described in "Oracle Database Vault Public Views" on page D-10.
new_value	New identity value, up to 1024 characters in mixed-case.

CREATE_DOMAIN_IDENTITY Function

This function adds an Oracle Real Application Clusters (RAC) database node to the domain factor identities and labels it according to the Oracle Label Security policy.

Syntax

```
CREATE_DOMAIN_IDENTITY(
  domain_name  VARCHAR2,
  domain_host  VARCHAR2,
  policy_name  VARCHAR2 DEFAULT NULL,
  domain_label VARCHAR2 DEFAULT NULL);
```

Parameters

Table E–37 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, use the DVF.F$DATABASE_DOMAIN view, described in "Oracle Database Vault PL/SQL Factor Functions" on page H-5.
domain_host	Oracle Real Application Clusters host name being added to the domain. To find host name of a database, use the DVF.F$DATABASE_HOSTNAME function, described in "Oracle Database Vault PL/SQL Factor Functions" on page H-5.
policy_name	Oracle Label Security policy name. To find the available policies, use the DBA_DV_MAC_POLICY view, described in "Oracle Database Vault Public Views" on page D-10.
domain_label	Name of the domain to which to add the Oracle Label Security policy.

CREATE_FACTOR Function

This function creates a factor. After you create a factor, you need to give it an identity by using the CREATE_IDENTITY function, described in "CREATE_IDENTITY Function" on page E-26.

Syntax

```
CREATE_FACTOR(
  factor_name       VARCHAR2,
  factor_type_name  VARCHAR2,
  description       VARCHAR2,
  rule_set_name     VARCHAR2,
  get_expr          VARCHAR2,
  validate_expr     VARCHAR2,
  identify_by       NUMBER,
  labeled_by        NUMBER,
  eval_options      NUMBER,
  audit_options     NUMBER,
  fail_options      NUMBER);
```

Parameters

Table E–38 CREATE_FACTOR Parameters

Parameter	Description
factor_name	Factor name, up to 30 characters in mixed-case, without spaces. To find existing factors in the current database instance, use the DBA_DV_FACTOR view, described in "Oracle Database Vault Public Views" on page D-10.
factor_type_name	Factor type name, up to 30 characters in mixed-case, without spaces.
description	Description of the purpose of the factor, up to 1024 characters in mixed-case.

Table E–38 (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, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10. See also "Assignment Rule Set" on page 7-8 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 "Retrieval Method" on page 7-7 for more information. See also the `audit_options` parameter.
`validate_expr`	Name of the function 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 "Validation Method" on page 7-7 for more information.
`identify_by`	Options for determining the identity of a factor, based on the expression set for the `get_expr` parameter: ■ `0`: By constant ■ `1`: By method ■ `2`: By factor See "Factor Identification" on page 7-5 for more information.
`labeled_by`	Options for 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. See "Factor Labeling" on page 7-6 for more information.
`eval_options`	Options for evaluating the factor when the user logs on: ■ `0`: When the database session is created ■ `1`: Each time the factor is accessed ■ `2`: On start-up See "Evaluation" on page 7-6 for more information.
`audit_options`	Options for auditing the factor if you want to generate a custom Oracle Database Vault audit record. ■ `0`: Disables auditing. ■ `POWER(2,0)`: Always audits. ■ `POWER(2,1)`: Audits if `get_expr` returns an error. ■ `POWER(2,2)`: Audits if `get_expr` is null. ■ `POWER(2,3)`: Audits if the validation function returns an error. ■ `POWER(2,4)`: Audits if the validation function is false. ■ `POWER(2,5)`: Audits if there is no trust level set. ■ `POWER(2,6)`: Audits if the trust level is negative. See "Audit Options" on page 7-8 for more information.
`fail_options`	Options for reporting factor errors: ■ `POWER(2,0)`: Shows an error message. ■ `POWER(2,1)`: Does not show an error message. See "Error Options" on page 7-9 for more information.

CREATE_FACTOR_TYPE Function

This function creates a user-defined factor type.

Syntax

```
CREATE_FACTOR_TYPE(
  name        VARCHAR2,
  description VARCHAR2);
```

Parameters

Table E–39 CREATE_FACTOR_TYPE Parameters

Parameter	Description
`name`	Factor type name, up to 30 characters in mixed-case, without spaces. To find existing factor types, use the `DBA_DV_FACTOR_TYPE` view, described in "Oracle Database Vault Public Views" on page D-10.
`description`	Description of the purpose of the factor type, up to 1024 characters in mixed-case.

CREATE_IDENTITY Function

This function assigns an identity and an associated trust level for a given factor. After you create a factor, you must assign it an identity.

Syntax

```
CREATE_IDENTITY(
  factor_name  VARCHAR2,
  value        VARCHAR2,
  trust_level  NUMBER);
```

Parameters

Table E–40 CREATE_IDENTITY Parameters

Parameter	Description
`factor_name`	Factor name. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`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 234.43.41.99.
`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 an Identity" on page 7-10 for more information about trust levels and label security.

CREATE_IDENTITY_MAP Function

This function defines a set of tests that are used to derive the identity of a factor from the value of linked child factors (subfactors).

Syntax

```
CREATE_IDENTITY_MAP(
  identity_factor_name  VARCHAR2,
  identity_factor_value VARCHAR2,
  parent_factor_name    VARCHAR2,
  child_factor_name     VARCHAR2,
  operation             VARCHAR2,
  operand1              VARCHAR2,
  operand2              VARCHAR2);
```

Parameters

Table E–41 CREATE_IDENTITY_MAP Parameters

Parameter	Description
`identity_factor_name`	Factor the identity map is for. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`identity_factor_value`	Value the factor will assume if the identity map evaluates to TRUE. To find existing factor identities, use the `DBA_DV_IDENTITY` view. To find current factor identity mappings, use `DBA_DV_IDENTITY_MAP`. Both are described in "Oracle Database Vault Public Views" on page D-10.
`parent_factor_name`	The parent factor link to which the map is related. To find existing parent-child factor mappings, use the `DBA_DV_IDENTITY_MAP` view, described in "Oracle Database Vault Public Views" on page D-10.
`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.

DELETE_FACTOR Function

This function deletes a factor.

Syntax

```
DELETE_FACTOR(
factor_name VARCHAR2);
```

Parameters

Table E–42 DELETE_FACTOR Parameter

Parameter	Description
`factor_name`	Factor name. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_FACTOR_LINK Function

This function removes a parent-child relationship for two factors.

Syntax

```
DELETE_FACTOR_LINK(
  parent_factor_name VARCHAR2,
  child_factor_name  VARCHAR2);
```

Parameters

Table E–43 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, use the `DBA_DV_FACTOR_LINK` view, described in "Oracle Database Vault Public Views" on page D-10.
`child_factor_name`	Factor name.

DELETE_FACTOR_TYPE Function

This function deletes a factor type.

Syntax

```
DELETE_FACTOR_TYPE(
  name VARCHAR2);
```

Parameters

Table E–44 DELETE_FACTOR_TYPE Parameters

Parameter	Description
`name`	Factor type name. To find existing factor types in the current database instance, use the `DBA_DV_FACTOR_TYPE` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_IDENTITY Function

This function removes an identity from an existing factor.

Syntax

```
DELETE_IDENTITY(
  factor_name VARCHAR2,
  value       VARCHAR2);
```

Parameters

Table E–45 DELETE_IDENTITY Parameters

Parameter	Description
`factor_name`	Factor name. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`value`	Identity value associated with the factor. To find the identities for each factor in the current database instance, use the `DBA_DV_IDENTITY` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_IDENTITY_MAP Function

This function removes an identity map for a factor.

Syntax

```
DELETE_IDENTITY_MAP(
  identity_factor_name  VARCHAR2,
  identity_factor_value VARCHAR2,
  parent_factor_name    VARCHAR2,
  child_factor_name     VARCHAR2,
  operation             VARCHAR2,
  operand1              VARCHAR2,
  operand2              VARCHAR2);
```

Parameters

Table E–46 DELETE_IDENTITY_MAP Parameters

Parameter	Description
`identity_factor_name`	Factor the identity map is for. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`identity_factor_value`	Value the factor will assume if the identity map evaluates to TRUE. To find existing factor identities, use the `DBA_DV_IDENTITY` view. To find current factor identity mappings, use `DBA_DV_IDENTITY_MAP`. Both are described in "Oracle Database Vault Public Views" on page D-10.
`parent_factor_name`	The parent factor link to which the map is related. To find existing factors, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`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.
`operand2`	Right operand for the relational operator.

DROP_DOMAIN_IDENTITY Function

This function removes an Oracle Real Application Clusters database node from a domain.

Syntax

```
DROP_DOMAIN_IDENTITY(
  domain_name  VARCHAR2,
  domain_host  VARCHAR2);
```

Parameters

Table E–47 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, use the `DVF.F$DATABASE_DOMAIN` function, described in "Oracle Database Vault PL/SQL Factor Functions" on page H-5.
`domain_host`	Oracle Real Application Clusters host name being that was added to the domain. To find the host name for a specified database, use the `DVF.F$DATABASE_HOSTNAME` function, described in "Oracle Database Vault PL/SQL Factor Functions" on page H-5.

GET_INSTANCE_INFO Function

This function returns information from the `SYS.V_$INSTANCE` view; it returns a `VARCHAR2` value. For more information about `SYS.V_$INSTANCE`, see *Oracle Database Reference*.

Syntax

```
GET_INSTANCE_INFO(
  p_parameter VARCHAR2);
```

Parameters

Table E–48 GET_INSTANCE_INFO Parameter

Parameter	Description
`p_parameter`	Column name in the `SYS.V_$INSTANCE` view. See *Oracle Database Reference* for a listing of the `SYS.V_$INSTANCE` columns.

GET_SESSION_INFO Function

This function returns information from the `SYS.V_$SESSION` view for the current session; it returns a `VARCHAR2` value. For more information about `SYS.V_$SESSION`, see *Oracle Database Reference*.

Syntax

```
GET_SESSION_INFO(
  p_parameter VARCHAR2);
```

Parameters

Table E–49 GET_SESSION_INFO Parameter

Parameter	Description
`p_parameter`	Column name in the `SYS.V_$SESSION` view.See *Oracle Database Reference* for a listing of the `SYS.V_$SESSION` columns.

RENAME_FACTOR Function

This function renames a factor. The name change takes effect everywhere the factor is used.

Syntax

```
RENAME_FACTOR(
  factor_name     VARCHAR2,
  new_factor_name VARCHAR2);
```

Parameters

Table E–50 RENAME_FACTOR Parameters

Parameter	Description
`factor_name`	Factor name. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`new_factor_name`	New factor name, up to 30 characters in mixed-case, without spaces.

RENAME_FACTOR_TYPE Function

This function renames a factor type. The name change takes effect everywhere the factor type is used.

Syntax

```
RENAME_FACTOR_TYPE(
  old_name  VARCHAR2,
  new_name  VARCHAR2);
```

Parameters

Table E–51 RENAME_FACTOR_TYPE Parameters

Parameter	Description
`old_name`	Current factor type name. To find existing factor types in the current database instance, use the `DBA_DV_FACTOR_TYPE` view, described in "Oracle Database Vault Public Views" on page D-10.
`new_name`	New factor type name, up to 30 characters in mixed-case, without spaces.

UPDATE_FACTOR Function

This function updates the description of a factor type.

Syntax

```
UPDATE_FACTOR(
  factor_name       VARCHAR2,
  factor_type_name  VARCHAR2,
  description       VARCHAR2,
  rule_set_name     VARCHAR2,
  get_expr          VARCHAR2,
  validate_expr     VARCHAR2,
  identify_by       NUMBER,
  labeled_by        NUMBER,
  eval_options      NUMBER,
  audit_options     NUMBER,
  fail_options      NUMBER);
```

Parameters

Table E–52 UPDATE_FACTOR

Parameter	Description
`factor_name`	Factor name. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`factor_type_name`	Factor type name. To find existing factor types, use the `DBA_DV_FACTOR_TYPE` view, described in "Oracle Database Vault Public Views" on page D-10.
`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, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10. See also "Assignment Rule Set" on page 7-8 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 "Retrieval Method" on page 7-7 for more information. See also the `audit_options` parameter.
`validate_expr`	Name of the function 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 "Validation Method" on page 7-7 for more information.
`identify_by`	Options for determining the identity of a factor, based on the expression set for the `get_expr` parameter: ■ `0`: By constant ■ `1`: By method ■ `2`: By factor ■ `3`: By context See "Factor Identification" on page 7-5 for more information.

Table E–52 (Cont.) UPDATE_FACTOR

Parameter	Description
`labeled_by`	Options for 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. See "Factor Labeling" on page 7-6 for more information.
`eval_options`	Options for evaluating the factor when the user logs on: ■ `0`: When the database session is created ■ `1`: Each time the factor is accessed ■ `2`: On start-up See "Evaluation" on page 7-6 for more information.
`audit_options`	Options for auditing the factor if you want to generate a custom Oracle Database Vault audit record. ■ `0`: Disables auditing. ■ `POWER(2,0)`: Always audits. ■ `POWER(2,1)`: Audits if `get_expr` returns an error. ■ `POWER(2,2)`: Audits if `get_expr` is null. ■ `POWER(2,3)`: Audits if the validation function returns an error. ■ `POWER(2,4)`: Audits if the validation function is false. ■ `POWER(2,5)`: Audits if there is no trust level set. ■ `POWER(2,6)`: Audits if the trust level is negative. See "Audit Options" on page 7-8 for more information.
`fail_options`	Options for reporting factor errors: ■ `POWER(2,0)`: Shows an error message. ■ `POWER(2,1)`: Does not show an error message. See "Error Options" on page 7-9 for more information.

UPDATE_FACTOR_TYPE Function

This function updates a factor type.

Syntax

```
UPDATE_FACTOR_TYPE(
  name         VARCHAR2,
  description  VARCHAR2);
```

Parameters

Table E–53 UPDATE_FACTOR_TYPE Parameters

Parameter	Description
`name`	Factor type name. To find existing factor types in the current database instance, use the `DBA_DV_FACTOR_TYPE` view, described in "Oracle Database Vault Public Views" on page D-10.

Table E–53 (Cont.) UPDATE_FACTOR_TYPE Parameters

Parameter	Description
description	Description of the purpose of the factor type, up to 1024 characters in mixed-case.

UPDATE_IDENTITY Function

This function updates the trust level of a factor identity.

Syntax

```
UPDATE_IDENTITY(
  factor_name  VARCHAR2,
  value        VARCHAR2,
  trust_level  NUMBER);
```

Parameters

Table E–54 UPDATE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, use the DBA_DV_FACTOR view, described in "Oracle Database Vault Public Views" on page D-10.
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 234.43.41.99.
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 an Identity" on page 7-10 for more information about trust levels and label security.

Secure Application Role Functions Within DVSYS.DBMS_MACADM

Table E–55 lists functions within the DVSYS.DBMS_MACADM package that you can use to configure Oracle Database Vault secure application roles.

Chapter 8, "Configuring Secure Application Roles for Oracle Database Vault" describes secure application roles in detail. See also Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility functions that you can use with the secure application role functions.

Table E–55 DVSYS.DBMS_MACADM Secure Application Role Configuration Functions

Function	Description
CREATE_ROLE Function	Creates an Oracle Database Vault secure application role.
DELETE_ROLE Function	Deletes an Oracle Database Vault secure application role.
RENAME_ROLE Function	Renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.
UPDATE_ROLE Function	Updates a Oracle Database Vault secure application role.

CREATE_ROLE Function

This function creates an Oracle Database Vault secure application role.

Syntax

```
CREATE_ROLE(
  role_name      VARCHAR2,
  enabled        VARCHAR2,
  rule_set_name  VARCHAR2);
```

Parameters

Table E–56 CREATE_ROLE Parameters

Parameter	Description
`role_name`	Role name, up to 30 characters, with no spaces. Preferably, enter the role name in upper case letters, though you are not required to do so. Ensure that this name follows the standard Oracle naming conventions for role creation described in *Oracle Database SQL Language Reference*. To find existing secure application roles in the current database instance, use the `DBA_DV_ROLE` view, described in "Oracle Database Vault Public Views" on page D-10.
`enabled`	`Y` (yes) enables role checking; `N` (no) disables it. The default is `Y`.
`rule_set_name`	Name of rule set to determine whether a user can set this secure application role. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_ROLE Function

This function deletes an Oracle Database Vault secure application role.

Syntax

```
DELETE_ROLE(
  role_name VARCHAR2);
```

Parameters

Table E–57 DELETE_ROLE Parameter

Parameter	Description
`role_name`	Role name. To find existing secure application roles in the current database instance, use the `DBA_DV_ROLE` view, described in "Oracle Database Vault Public Views" on page D-10.

RENAME_ROLE Function

This function renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.

Syntax

```
RENAME_ROLE(
  role_name      VARCHAR2,
  new_role_name  VARCHAR2);
```

Parameters

Table E–58 RENAME_ROLE Parameters

Parameter	Description
`role_name`	Role name. To find existing secure application roles in the current database instance, use the `DBA_DV_ROLE` view, described in "Oracle Database Vault Public Views" on page D-10.
`new_role_name`	Role name, up to 30 characters, in uppercase, with no spaces. Ensure that this name follows the standard Oracle naming conventions for role creation described in *Oracle Database SQL Language Reference*.

UPDATE_ROLE Function

This function updates a Oracle Database Vault secure application role.

Syntax

```
UPDATE_ROLE(
  role_name      VARCHAR2,
  enabled        VARCHAR2,
  rule_set_name  VARCHAR2);
```

Parameters

Table E–59 UPDATE_ROLE Parameters

Parameter	Description
`role_name`	Role name. To find existing secure application roles in the current database instance, use the `DBA_DV_ROLE` view, described in "Oracle Database Vault Public Views" on page D-10.
`enabled`	`Y` (yes) enables role checking; `N` (no) disables it. The default is `Y`.
`rule_set_name`	Name of rule set to determine whether a user can set this secure application role. To find existing rule sets in the current database instance, use the `DBA_DV_RULE_SET` view, described in "Oracle Database Vault Public Views" on page D-10.

Oracle Label Security Policy Functions Within DVSYS.DBMS_MACADM

Table E–60 lists functions within the `DVSYS.DBMS_MACADM` package that you can use to configure Oracle Label Security policies.

Chapter 9, "Integrating Oracle Database Vault with Other Oracle Products" describes Oracle Label Security policies in detail. See also Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility functions that you can use with the Oracle Label Security policy functions.

Table E–60 DVSYS.DBMS_MACADM Oracle Label Security Configuration Functions

Function	Description
CREATE_MAC_POLICY Function	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 Function	Labels an identity within an Oracle Label Security policy.
DELETE_MAC_POLICY_CASCADE Function	Deletes all Oracle Database Vault objects related to an Oracle Label Security policy.
DELETE_POLICY_FACTOR Function	Removes the factor from contributing to the Oracle Label Security label.
DELETE_POLICY_LABEL Function	Removes the label from an identity within an Oracle Label Security policy.
UPDATE_MAC_POLICY Function	Specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

CREATE_MAC_POLICY Function

This function specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Syntax

```
CREATE_MAC_POLICY(
  policy_name  VARCHAR2,
  algorithm    VARCHAR2);
```

Parameters

Table E–61 CREATE_MAC_POLICY Parameters

Parameter	Description
`policy_name`	Name of existing policy. To find existing policies in the current database instance, use the `DBA_DV_MAC_POLICY` view, described in "Oracle Database Vault Public Views" on page D-10.
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. Enter the code listed in Table E–62 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 E–62 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

Table E–62 (Cont.) Merge Algorithm Codes

Code	Value
`HII`	Maximum Level/Intersection/Intersection
`HMI`	Maximum Level/Minus/Intersection
`HNI`	Maximum Level/Null/Intersection
`HUM`	Maximum Level/Union/Minus
`HIM`	Maximum Level/Intersection/Minus
`HMM`	Maximum Level/Minus/Minus
`HNM`	Maximum Level/Null/Minus
`HUN`	Maximum Level/Union/Null
`HIN`	Maximum Level/Intersection/Null
`HMN`	Maximum Level/Minus/Null
`HNN`	Maximum Level/Null/Null
`LUU`	Minimum Level/Union/Union
`LIU`	Minimum Level/Intersection/Union
`LMU`	Minimum Level/Minus/Union
`LNU`	Minimum Level/Null/Union
`LUI`	Minimum Level/Union/Intersection
`LII`	Minimum Level/Intersection/Intersection
`LMI`	Minimum Level/Minus/Intersection
`LNI`	Minimum Level/Null/Intersection
`LUM`	Minimum Level/Union/Minus
`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

CREATE_POLICY_LABEL Function

This function labels an identity within an Oracle Label Security policy.

Syntax

```
CREATE_POLICY_LABEL(
  identity_factor_name   VARCHAR2,
  identity_factor_value  VARCHAR2,
  policy_name            VARCHAR2,
  label                  VARCHAR2);
```

Parameters

Table E–63 CREATE_POLICY_LABEL Parameters

Parameter	Description
`identity_factor_name`	Name of factor being labeled. To find existing factors in the current database instance, use the `DBA_DV_FACTOR` view. To find factors that are associated with Oracle Label Security policies, use `DBA_DV_MAC_POLICY_FACTOR`. Both are described in "Oracle Database Vault Public Views" on page D-10. See also "Label Security Policy Factors" on page 9-5 for more information.
`identity_factor_value`	Value of identity for the factor being labeled. To find the identities of existing factors in the current database instance, use the `DBA_DV_IDENTITY` view, described in "Oracle Database Vault Public Views" on page D-10.
`policy_name`	Name of existing policy. To find existing policies in the current database instance, use the `DBA_DV_MAC_POLICY` view, described in "Oracle Database Vault Public Views" on page D-10.
`label`	Oracle Label Security label name. To find existing policy labels for factor identifiers, use the `DBA_DV_POLICY_LABEL` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_MAC_POLICY_CASCADE Function

This function deletes all Oracle Database Vault objects related to an Oracle Label Security policy.

Syntax

```
DELETE_MAC_POLICY_CASCADE(
  policy_name  VARCHAR2);
```

Parameters

Table E–64 DELETE_MAC_POLICY_CASCADE Parameter

Parameter	Description
`policy_name`	Name of existing policy. To find existing policies in the current database instance, use the `DBA_DV_MAC_POLICY` view, described in "Oracle Database Vault Public Views" on page D-10.

DELETE_POLICY_FACTOR Function

This function removes the factor from contributing to the Oracle Label Security label.

Syntax

```
DELETE_POLICY_FACTOR(
  policy_name  VARCHAR2,
  factor_name  VARCHAR2);
```

Parameters

Table E–65 DELETE_POLICY_FACTOR Parameters

Parameter	Description
`policy_name`	Name of existing policy. To find existing policies in the current database instance, use the `DBA_DV_MAC_POLICY` view, described in "Oracle Database Vault Public Views" on page D-10.
`factor_name`	Name of factor associated with the Oracle Label Security label. To find factors that are associated with Oracle Label Security policies, use `DBA_DV_MAC_POLICY_FACTOR`. Both are described in "Oracle Database Vault Public Views" on page D-10.

DELETE_POLICY_LABEL Function

This function removes the label from an identity within an Oracle Label Security policy.

Syntax

```
DELETE_POLICY_LABEL(
  identity_factor_name   VARCHAR2,
  identity_factor_value  VARCHAR2,
  policy_name            VARCHAR2,
  label                  VARCHAR2);
```

Parameters

Table E–66 DELETE_POLICY_LABEL Parameters

Parameter	Description
`identity_factor_name`	Name of factor that was labeled. To find existing factors in the current database instance that are associated with Oracle Label Security policies, use `DBA_DV_MAC_POLICY_FACTOR`. Both are described in "Oracle Database Vault Public Views" on page D-10. See also "Label Security Policy Factors" on page 9-5 for more information.
`identity_factor_value`	Value of identity for the factor that was labeled. To find the identities of existing factors in the current database instance, use the `DBA_DV_IDENTITY` view, described in "Oracle Database Vault Public Views" on page D-10.
`policy_name`	Name of existing policy. To find existing policies in the current database instance, use the `DBA_DV_MAC_POLICY` view, described in "Oracle Database Vault Public Views" on page D-10.
`label`	Oracle Label Security label name. To find existing policy labels for factor identifiers, use the `DBA_DV_POLICY_LABEL` view, described in "Oracle Database Vault Public Views" on page D-10.

UPDATE_MAC_POLICY Function

This function specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Syntax

```
UPDATE_MAC_POLICY(
  policy_name  VARCHAR2,
  algorithm    VARCHAR2);
```

Parameters

Table E–67 UPDATE_MAC_POLICY

Parameter	Description
`policy_name`	Name of existing policy. To find existing policies in the current database instance, use the `DBA_DV_MAC_POLICY` view, described in "Oracle Database Vault Public Views" on page D-10.
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. For example: `"LII - Minimum Level/Intersection/Intersection"` For more information on label-merging algorithms, see *Oracle Label Security Administrator's Guide*.

F

Oracle Database Vault DVSYS.DBMS_MACSEC_ROLES Package

You can modify your applications to use the functions within the DVSYS.DBMS_MACSEC_ROLES package to check the authorization for a user or to set an Oracle Database Vault secure application role. The DVSYS.DBMS_MACSEC_ROLES package is available to all users.

Chapter 8, "Configuring Secure Application Roles for Oracle Database Vault" describes secure application roles in detail. See also Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility functions that you can use with the secure application role functions.

Table F–1 lists the DVSYS.DBMS_MACSEC_ROLES package functions.

Table F–1 DVS.DBMS_MACSEC_ROLES Oracle Label Security Configuration Functions

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 Function	Issues the SET ROLE statement for an Oracle Database Vault secure application role.

CAN_SET_ROLE Function

This function checks whether the user invoking the method is authorized to use the specified Oracle Database Vault secure application role. It returns a BOOLEAN value.

Syntax

```
CAN_SET_ROLE(
  p_role VARCHAR2);
```

Parameters

Table F–2 CAN_SET_ROLE Parameter

Parameter	Description
p_role	Role name. To find existing secure application roles, use the DBA_DV_ROLE view, described in "Oracle Database Vault Public Views" on page D-10.

SET_ROLE Function

This function issues the SET ROLE statement for an Oracle Database Vault secure application role. If a rule set that is associated with the role evaluates to false, then the role is not set.

Syntax

```
SET_ROLE(
  p_role VARCHAR2);
```

Parameters

Table F–3 SET_ROLE Parameter

Parameter	Description
p_role	Role name. To find existing secure application roles in the current database instance, use the DBA_DV_ROLE view, described in "Oracle Database Vault Public Views" on page D-10.

G

Oracle Database Vault DVSYS.DBMS_MACUTL Package

The DVSYS.DBMS_MACUTL package provides a set of general purpose utility functions that you can use throughout the application code you write for Oracle Database Vault. This package is available to all users.

This chapter includes the following topics:

- Field Summary
- Functions Within the DVSYS.DBMS_MACUTL Package

Field Summary

Table G–1 summarizes field (that is, constant) descriptions for the DVSYS.DBMS_MACUTL package.

Table G–1 DVSYS.DBMS_MACUTL Field Summary

FIeld Name	Data Type	Description
G_ALL_OBJECT	VARCHAR2(1)	Realm Objects: Wildcard to indicate all object names or all object types
G_AUDIT_ALWAYS	NUMBER	Factor audit_options: Always audit.
G_AUDIT_OFF	NUMBER	Factor audit_options: No audit.
G_AUDIT_ON_GET_ERROR	NUMBER	Factor audit_options: Audit if get_expr returns an error.
G_AUDIT_ON_GET_NULL	NUMBER	Factor audit_options: Audit if get_expr is null.
G_AUDIT_ON_TRUST_LEVEL_NEG	NUMBER	Factor audit_options: Audit if trust level is negative.
G_AUDIT_ON_TRUST_LEVEL_NULL	NUMBER	Factor audit_options: Audit if no trust level exists.
G_AUDIT_ON_VALIDATE_ERROR	NUMBER	Factor audit_options: Audit if validation function returns an error.
G_AUDIT_ON_VALIDATE_FALSE	NUMBER	Factor audit_options: Audit if validation function is false.
G_CODES_AUDIT_EVENTS	VARCHAR2(30)	Code groups: Audit event descriptions
G_CODES_BOOLEAN	VARCHAR2(30)	Code groups: BOOLEAN values

Table G–1 (Cont.) DVSYS.DBMS_MACUTL Field Summary

FIeld Name	Data Type	Description
`G_CODES_DB_OBJECT_TYPE`	`VARCHAR2(30)`	Code groups: Database object types
`G_CODES_DDL_CMDS`	`VARCHAR2(30)`	Code groups: DDL commands.
`G_CODES_FACTOR_AUDIT`	`VARCHAR2(30)`	Code groups: Factor `audit_options`.
`G_CODES_FACTOR_EVAL`	`VARCHAR2(30)`	Code groups: Factor `eval_options`
`G_CODES_FACTOR_FAIL`	`VARCHAR2(30)`	Code groups: Factor `fail_options`
`G_CODES_FACTOR_IDENTIFY`	`VARCHAR2(30)`	Code groups: Factor `identity_by`
`G_CODES_FACTOR_LABEL`	`VARCHAR2(30)`	Code groups: Factor `labeled_by`
`G_CODES_LABEL_ALG`	`VARCHAR(30)`	Code groups: Oracle Label Security Policy merge algorithms
`G_CODES_MESSAGES`	`VARCHAR(30)`	Code groups: Oracle Database Vault Error messages
`G_CODES_OPERATORS`	`VARCHAR2(30)`	Code groups: SQL relational operators
`G_CODES_REALM_AUDIT`	`VARCHAR2(30)`	Code groups: Realm `audit_options`
`G_CODES_RULESET_AUDIT`	`VARCHAR2(30)`	Code groups: Rule Set `audit_options`
`G_CODES_RULESET_EVAL`	`VARCHAR2(30)`	Code groups: Rule set `evaluate_options`
`G_CODES_RULESET_EVENT`	`VARCHAR2(30)`	Code groups: Rule set handler_options
`G_CODES_RULESET_FAIL`	`VARCHAR(30)`	Code groups: Rule set `fail_options`
`G_CODES_SQL_CMDS`	`VARCHAR2(30)`	Code groups: SQL statements
`G_CONTEXT_FACTOR`	`VARCHAR2(30)`	Factors: `MAC$FACTOR`, *`factor name`*, *`factor value`* The term *context* in the field name refers to the application context capability in Oracle Database.
`G_CONTEXT_FACTOR_LABEL`	`VARCHAR2(30)`	Factor labels: `MAC$F$` *`policy`*, *`factor_name`*, *`factor label`* The term *context* in the field name refers to the application context capability in Oracle Database.
`G_CONTEXT_PREFIX`	`VARCHAR2(30)`	The access control and Oracle Label Security context start with this field name The term *context* in the field name refers to the application context capability in Oracle Database.

Table G–1 (Cont.) DVSYS.DBMS_MACUTL Field Summary

FIeld Name	Data Type	Description
G_CONTEXT_REALM	VARCHAR2(30)	Realm: MAC$REALM, *factor name*, *factor value* The term *context* in the field name refers to the application context capability in Oracle Database.
G_CONTEXT_SESSION_LABEL	VARCHAR2(30)	Session Labels: MAC$S$ *policy*, *session attribute*, *label* The term *context* in the field name refers to the application context capability in Oracle Database.
G_EVAL_ON_ACCESS	NUMBER	Factor eval_options: Reevaluate on each access
G_EVAL_ON_SESSION	NUMBER	Factor eval_options: Evaluate once upon login
G_EVAL_ON_STARTUP	NUMBER	Factor eval_options: Evaluate once at database startup
G_FAIL_SILENTLY	NUMBER	Fail_options: Fail with no message
G_FAIL_WITH_MESSAGE	NUMBER	Fail_options: Fail with message
G_IDENTIFY_BY_CONSTANT	NUMBER	Factor identify_by column: Fixed value in get_expr column
G_IDENTIFY_BY_CONTEXT	NUMBER	Factor identify_by context
G_IDENTIFY_BY_FACTOR	NUMBER	Factor identify_by column: Subfactors through factor_link$ table
G_IDENTIFY_BY_METHOD	NUMBER	Factor identify_by column: Expression in get_expr column
G_IDENTIFY_BY_RULESET	NUMBER	Factor identify_by column: Expression and Rule Set with factor_expr$ table
G_LABELED_BY_FACTORS	NUMBER	Factor labeled_by column: Derive label from subfactor and merge algorithm
G_LABELED_BY_SELF	NUMBER	Factor labeled_by column: Factor's identities are labeled
G_MAX_SESSION_LABEL	VARCHAR2(30)	This is the highest label a user could set based on the factors. It does not take into account the label for a user.
G_MIN_POLICY_LABEL	VARCHAR2(30)	The label that a factor with a null label defaults to
G_NO	VARCHAR2(1)	No constant for enabled and label_ind columns (BOOLEAN FALSE)
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	Realm audit_options: Audit on realm violation.

Table G–1 (Cont.) DVSYS.DBMS_MACUTL Field Summary

FIeld Name	Data Type	Description
G_REALM_AUDIT_OFF	NUMBER	Realm audit_options: No auditing
G_REALM_AUDIT_SUCCESS	NUMBER	Realm audit_options: Audit on successful realm access
G_REALM_AUTH_OWNER	NUMBER	Realm authorizations: Owner
G_REALM_AUTH_PARTICIPANT	NUMBER	Realm authorizations: Participant
G_RULESET_AUDIT_FAIL	NUMBER	Rule set audit_options: Audit on rule set failure
G_RULESET_AUDIT_OFF	NUMBER	Rule set audit_options: No auditing
G_RULESET_AUDIT_SUCCESS	NUMBER	Rule set audit_options: Audit on rule set success
G_RULESET_EVAL_ALL	NUMBER	Rule set eval_options: Rule set succeeds if all rules are TRUE
G_RULESET_EVAL_ANY	NUMBER	Rule set eval_options: Rule set succeeds if any rule is TRUE
G_RULESET_FAIL_SHOW	NUMBER	Rule set fail_options: Show error message
G_RULESET_FAIL_SILENT	NUMBER	Rule set fail_options: No error message
G_RULESET_HANDLER_FAIL	NUMBER	Rule set handler_options: Call handler on rule set failure
G_RULESET_HANDLER_OFF	NUMBER	Rule set handler_options: No call to handler
G_RULESET_HANDLER_SUCCESS	NUMBER	Rule set handler_options: Call handler on rule set success
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)	Yes constant for enabled and label_ind columns (BOOLEAN TRUE)

Functions Within the DVSYS.DBMS_MACUTL Package

Table G–2 lists the functions in the DVSYS.DBMS_MACUTL package.

Table G–2 DVSYS.DBMS_MACUTL Utility Functions

Function	Descriptions
CHECK_DVSYS_DML_ALLOWED Function	Verifies that public-packages are not being bypassed by users updating the Oracle Database Vault configuration.
GET_CODE_ID Function	Looks up the ID for a code within a code group; returns a NUMBER value.
GET_CODE_VALUE Function	Looks up the value for a code within a code group; returns a VARCHAR2 value.

Table G–2 (Cont.) DVSYS.DBMS_MACUTL Utility Functions

Function	**Descriptions**
GET_FACTOR_CONTEXT Function	Constructs an XML document that contains the values for all of the factors; returns a `VARCHAR2` value. Useful for rule expressions based on time data. Use this function to retrieve factors at the current time for a session. It is also useful for auditing purposes.
GET_SECOND Function	Returns the seconds in Oracle SS format (00-59); returns a `NUMBER` value. Useful for rule expressions based on time data.
GET_MINUTE Function	Returns the minute in Oracle MI format (00–59); returns a `NUMBER` value. Useful for rule expressions based on time data.
GET_HOUR Function	Returns the month in Oracle HH24 format (00–23); returns a `NUMBER` value. Useful for rule expressions based on time data.
GET_DAY Function	Returns the day in Oracle DD format (01–31); returns a `NUMBER` value. Useful for rule expressions based on time data.
GET_MONTH Function	Returns the month in Oracle MM format (01–12); returns a `NUMBER` value. Useful for rule expressions based on time data.
GET_YEAR Function	Returns the year in Oracle YYYY format (0001–9999); returns a `NUMBER` value. Useful for rule expressions based on time data.
IN_CALL_STACK Function	Checks for a string in the PL/SQL call stack; returns a `BOOLEAN` value.
GET_SQL_TEXT Function	Concatenates the elements of `ora_name_list_t` into a single `VARCHAR2` value; returns a `VARCHAR2` value.
IS_ALPHA Function	Checks whether the character is alphabetic; returns a `BOOLEAN` value.
IS_DIGIT Function	Checks whether the character is numeric; returns a `BOOLEAN` value.
IS_DVSYS_OWNER Function	Determines whether a user is authorized to manage the Oracle Database Vault configuration; returns a `BOOLEAN` value.
IS_OLS_INSTALLED Function	Returns an indicator as to whether or not Oracle Label Security is installed; returns a `BOOLEAN` value.
IS_OLS_INSTALLED_VARCHAR Function	Returns an indicator as to whether or not Oracle Label Security is installed; returns a `VARCHAR2` value.
RAISE_UNAUTHORIZED_OPERATION Function	Generates an ORA-20920 (Unauthorized Operation) error
GET_MESSAGE_LABEL Function	Looks up an Oracle RDBMS error message; returns a `VARCHAR2` value.
GET_MESSAGE_LABEL Function	Looks up an Oracle RDBMS error message; returns a `VARCHAR2` value.
TO_ORACLE_IDENTIFIER Function	Alters a string to make it a legal Oracle identifier; returns a `VARCHAR2` value.
USER_HAS_OBJECT_PRIVILEGE Function	Checks whether a user or role may access an object through an object privilege grant; returns a `BOOLEAN` value.
USER_HAS_ROLE Function	Checks whether a user has a role privilege, directly or indirectly (through another role); returns a `BOOLEAN` value.
USER_HAS_ROLE_VARCHAR Function	Checks whether a user has a role privilege, directly or indirectly (through another role); returns a `VARCHAR2` value.
USER_HAS_SYSTEM_PRIVILEGE Function	Checks whether a user has a system privilege, directly or indirectly (through a role); returns a `BOOLEAN` value.

CHECK_DVSYS_DML_ALLOWED Function

This function verifies that public packages are not being bypassed by users updating the Oracle Database Vault configuration.

Syntax

```
CHECK_DVSYS_DML_ALLOWED(
  p_user VARCHAR2 DEFAULT USER);
```

Parameter

Table G–3 CHECK_DVSYS_DML_ALLOWED Parameter

Parameter	Description
p_user	User performing the operation. To find existing users in the current database instance, use the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DVA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See "Oracle Database Vault Public Views" on page D-10. ■ DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "Oracle Database Vault Public Views" on page D-10.

GET_CODE_ID Function

This function looks up the ID for a code within a code group, and then returns a NUMBER value. These codes are used for the user interface, views, and for validating input in a translatable fashion.

Syntax

```
GET_CODE_ID(
  p_code_group VARCHAR2,
  p_code       VARCHAR2);
```

Parameters

Table G–4 GET_CODE_ID Parameters

Parameter	Description
p_code_group	Code group, for example, AUDIT_EVENTS or BOOLEAN. To find available code groups in the current database instance, use the DBA_DV_CODE view, described in "Oracle Database Vault Public Views" on page D-10.
p_code	Value of the code from DBA_DV_CODE. This value is listed when you run the DBA_DV_CODE view.

GET_CODE_VALUE Function

This function looks up the value for a code within a code group, and then returns a VARCHAR2 value.

Syntax

```
GET_CODE_VALUE(
  p_code_group VARCHAR2,
  p_code       VARCHAR2);
```

Parameters

Table G–5 GET_CODE_VALUE Parameters

Parameter	Description
p_code_group	Code group, for example, AUDIT_EVENTS or BOOLEAN. To find existing code groups in the current database instance, use the DBA_DV_CODE view, described in "Oracle Database Vault Public Views" on page D-10.
p_code	ID of the code. This ID is listed when you run the DBA_DV_CODE view.

GET_FACTOR_CONTEXT Function

This function constructs an XML document that contains the values for all of the factors. This XML document is only intended for auditing or tracing and is truncated if it is longer than 4000 characters. The function returns a VARCHAR2 value.

Use this function to retrieve factors at the current time for a session. It is also useful for auditing purposes.

Syntax

```
GET_FACTOR_CONTEXT();
```

Parameters

None.

GET_SECOND Function

This 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

```
GET_SECOND(
  p_date DATE DEFAULT SYSDATE);
```

Parameter

Table G–6 GET_SECOND Parameter

Parameter	Description
p_date	Date in SS format, for example: 59. If you do not specify a date, 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.

GET_MINUTE Function

This function returns the minute in Oracle MI (minute) format (00–59); returns a NUMBER value. Useful for rule expressions based on time data.

Syntax

```
GET_MINUTE(
  p_date DATE DEFAULT SYSDATE);
```

Parameter

Table G–7 GET_MINUTE Parameter

Parameter	Description
p_date	Date in MI format, for example, 30 (as is 2:30). If you do not specify a date, 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.

GET_HOUR Function

This function returns the hour in Oracle HH24 (hour) format (00–23); returns a NUMBER value. Useful for rule expressions based on time data.

Syntax

```
GET_HOUR(
  p_date DATE DEFAULT SYSDATE);
```

Parameter

Table G–8 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, 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.

GET_DAY Function

This function returns the day in Oracle DD (day) format (01–31); returns a NUMBER value. It is useful for rule expressions based on time data.

Syntax

```
GET_DAY(
  p_date DATE DEFAULT SYSDATE);
```

Parameter

Table G–9 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, 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.

GET_MONTH Function

This function returns the month in Oracle MM (month) format (01–12); returns a NUMBER value. Useful for rule expressions based on time data.

Syntax

```
GET_MONTH(
  p_date DATE DEFAULT SYSDATE);
```

Parameter

Table G–10 GET_MONTH Parameter

Parameter	Description
p_date	Date in MM format, for example, 08 for August. If you do not specify a date, 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.

GET_YEAR Function

This function returns the year in Oracle YYYY (year) format (0001–9999); returns a NUMBER value. Useful for rule expressions based on time data.

Syntax

```
GET_YEAR(
  p_date DATE DEFAULT SYSDATE);
```

Parameter

Table G–11 GET_YEAR Parameter

Parameter	Description
p_date	Date in YYYY format, for example, 1984. If you do not specify a date, Oracle Database Vault uses the SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

GET_SQL_TEXT Function

This function concatenates the elements of ora_name_list_t into a single VARCHAR2 value, and then returns a VARCHAR2 value.

Syntax

```
GET_SQL_TEXT(
  p_sql_text ora_name_list_t);
```

Parameters

Table G–12 GET_SQL_TEXT Parameter

Parameter	Description
p_sql_text	Table of VARCHAR2 strings representing SQL text, for example, SELECT, DROP TABLE, and so on.

IN_CALL_STACK Function

This function checks for a string in the PL/SQL call stack, and then returns a BOOLEAN value. IN_CALL_STACK returns TRUE if the string is in the call stack.

Syntax

```
IN_CALL_STACK(
  p_search_term VARCHAR2);
```

Parameter

Table G–13 IN_CALL_STACK Parameter

Parameter	Description
p_search_term	String to search for

IS_ALPHA Function

This function checks whether the character is alphabetic, and then returns a BOOLEAN value. IS_ALPHA returns TRUE if the character is alphabetic.

Syntax

```
IS_ALPHA(
  c VARCHAR2);
```

Parameter

Table G–14 IS_ALPHA Parameter

Parameter	Description
c	String with one character

IS_DIGIT Function

This function checks whether the character is numeric, and then returns a BOOLEAN value. IS_DIGIT returns TRUE if the character is a digit.

Syntax

```
IS_DIGIT(
  c VARCHAR2);
```

Parameter

Table G–15 IS_DIGIT Parameter

Parameter	Description
c	String with one character

IS_DVSYS_OWNER Function

This 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

```
IS_DVSYS_OWNER(
  p_user VARCHAR2 DEFAULT USER);
```

Parameter

Table G–16 IS_DVSYS_OWNER Parameter

Parameter	Description
p_user	User to check. To find existing users, use the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DVA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See "Oracle Database Vault Public Views" on page D-10. ■ DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "Oracle Database Vault Public Views" on page D-10.

IS_OLS_INSTALLED Function

This function returns an indicator as to whether or not 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

```
IS_OLS_INSTALLED()
```

Parameters

None.

IS_OLS_INSTALLED_VARCHAR Function

This function returns an indicator as to whether or not Oracle Label Security is installed, and then returns a Y or N VARCHAR2 value. If Oracle Label Security is installed, IS_OLS_INSTALLED_VARCHAR returns Y.

Syntax

```
IS_OLS_INSTALLED_VARCHAR()
```

Parameters

None.

GET_MESSAGE_LABEL Function

This function looks up an Oracle RDBMS error message, and then returns a VARCHAR2 value.

Syntax

```
GET_MESSAGE_LABEL(
  p_message_code VARCHAR2);
```

Parameters

Table G–17 GET_MESSAGE_LABEL Parameter

Parameter	Description
`p_message`	Message code. See *Oracle Database Error Messages* for a listing of error messages.
`p_parameter1`	Value to substitute for %1
`p_parameter2`	Value to substitute for %2
`p_parameter3`	Value to substitute for %3
`p_parameter4`	Value to substitute for %4
`p_parameter5`	Value to substitute for %5
`p_parameter6`	Value to substitute for %6

GET_MESSAGE_LABEL Function

This function looks up an Oracle RDBMS error message, and then returns a `NUMBER` value.

Syntax

```
GET_MESSAGE_LABEL(
  p_message_code NUMBER);
```

Parameters

Table G–18 GET_MESSAGE_LABEL Parameter

Parameter	Description
`p_message`	Message code. See *Oracle Database Error Messages* for a listing of error messages.
`p_parameter1`	Value to substitute for %1
`p_parameter2`	Value to substitute for %2
`p_parameter3`	Value to substitute for %3
`p_parameter4`	Value to substitute for %4
`p_parameter5`	Value to substitute for %5
`p_parameter6`	Value to substitute for %6

RAISE_UNAUTHORIZED_OPERATION Function

This function generates an ORA-20920 (Unauthorized Operation) error for unauthorized users.

Syntax

```
RAISE_UNAUTHORIZED_OPERATION(
  p_user VARCHAR2 DEFAULT USER);
```

Parameter

Table G–19 RAISE_UNAUTHORIZED_OPERATION Parameter

Parameter	Description
`p_user`	User performing the operation. To find existing users, use the following views: ■ `DBA_USERS`: Finds available users for the current database instance. See *Oracle Database Reference*. ■ `DVA_DV_REALM_AUTH`: Finds the authorization of a particular user or role. See "Oracle Database Vault Public Views" on page D-10. ■ `DBA_DV_ROLE`: Finds existing secure application roles used in privilege management. See "Oracle Database Vault Public Views" on page D-10.

TO_ORACLE_IDENTIFIER Function

This function turns string into a legal Oracle identifier, and then returns a `VARCHAR2` value.

Syntax

```
TO_ORACLE_IDENTIFIER(
  id VARCHAR2);
```

Parameter

Table G–20 TO_ORACLE_IDENTIFIER Parameter

Parameter	Description
`id`	Illegal identifier

USER_HAS_OBJECT_PRIVILEGE Function

This function checks whether a user or role may access an object through an object privilege grant, and then returns a `BOOLEAN` value. If the user or role has object privileges, then `USER_HAS_OBJECT_PRIVILEGE` returns TRUE.

Syntax

```
USER_HAS_OBJECT_PRIVILEGE(
  p_user         VARCHAR2,
  p_object_owner  VARCHAR2,
  p_object_name   VARCHAR2,
  p_privilege     VARCHAR2);
```

Parameters

Table G–21 USER_HAS_OBJECT_PRIVILEGE Parameters

Parameter	Description
`p_user`	User or role to check. To find existing users, use the 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 "Oracle Database Vault Public Views" on page D-10. ■ `DBA_DV_ROLE`: Finds existing secure application roles used in privilege management. See "Oracle Database Vault Public Views" on page D-10.
`p_object_owner`	Object owner. To find the available users, use the `DBA_USERS` view, described in *Oracle Database Reference*. To find the authorization of a particular user, use the `DVA_DV_REALM_AUTH` view, described in "Oracle Database Vault Public Views" on page D-10.
`p_object_name`	Object name. To find the available objects, use the `ALL_OBJECTS` view, described in *Oracle Database Reference*. To find objects that are secured by existing realms, use the `DBA_DV_REALM_OBJECT` view, described in "Oracle Database Vault Public Views" on page D-10.
`p_privilege`	Object privilege, for example, `SELECT`, `UPDATE`, `INSERT`, and so on. To find privileges for a database account excluding `PUBLIC` privileges, use the `DBA_DV_USER_PRIVS` view. To find all privileges for a database account, use `DBA_DV_USER_PRIVS_ALL`. Both are described in "Oracle Database Vault Public Views" on page D-10.

USER_HAS_ROLE Function

This 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

```
USER_HAS_ROLE(
  p_role VARCHAR2,
  p_user VARCHAR2 DEFAULT USER);
```

Parameters

Table G–22 USER_HAS_ROLE Parameters

Parameter	Description
p_role	Role privilege to check. To find existing roles, use the following views: ■ 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 "Oracle Database Vault Public Views" on page D-10. ■ DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "Oracle Database Vault Public Views" on page D-10.
p_user	User to check. To find existing users, use the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DVA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See "Oracle Database Vault Public Views" on page D-10.

USER_HAS_ROLE_VARCHAR Function

This 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

```
USER_HAS_ROLE_VARCHAR(
  p_role VARCHAR2,
  p_user VARCHAR2 DEFAULT USER);
```

Parameters

Table G–23 USER_HAS_ROLE_VARCHAR Parameters

Parameter	Description
p_role	Role to check. To find existing roles, use the following views: ■ 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 "Oracle Database Vault Public Views" on page D-10. ■ DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "Oracle Database Vault Public Views" on page D-10.
p_user	User to check. To find existing users, use the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DVA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See "Oracle Database Vault Public Views" on page D-10.

USER_HAS_SYSTEM_PRIVILEGE Function

This 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

```
USER_HAS_SYSTEM_PRIVILEGE(
  p_privilege  VARCHAR2,
  p_user       VARCHAR2 DEFAULT USER);
```

Parameters

Table G–24 USER_HAS_SYSTEM_PRIVILEGE Parameters

Parameter	Description
p_privilege	System privilege to check for. To find privileges for a database account excluding PUBLIC privileges, use the DBA_DV_USER_PRIVS view. To find all privileges for a database account, use DBA_DV_USER_PRIVS_ALL. Both are described in "Oracle Database Vault Public Views" on page D-10.
p_user	User to check. To find existing users, use the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DVA_DV_REALM_AUTH: Finds the authorization of a particular user or role. See "Oracle Database Vault Public Views" on page D-10.

H

PL/SQL Interfaces to Oracle Database Vault

This appendix includes the following sections:

- Oracle Database Vault Run-Time PL/SQL Procedures and Functions
- Oracle Database Vault PL/SQL Factor Functions
- Oracle Database Vault PL/SQL Rule Functions
- Oracle Database Vault PL/SQL Packages

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 `EXECUTE` privileges 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 H–1 lists the procedures and functions that are used to enable Oracle Database Vault processing with the `DVSYS` schema.

Table H–1 DVSYS Functions

Function	Parameter
SET_FACTOR Function	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 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 Function

This function can be exposed to an application that requires the ability to set factor identities dynamically. It wraps the package procedure DBMS_MACSEC.SET_FACTOR. When a factor has a rule set associated with it for assignment and if the rule set returns true, then the value will be set. Normal rule set handling occurs, and the factor value (identity) validation method will be called. This procedure is available (to execute) to the general database account population.

Syntax

```
SET_FACTOR(
  p_factor VARCHAR2,
  p_value  VARCHAR2);
```

Parameters

Table H–2 SET_FACTOR Parameters

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, use the DBA_DV_FACTOR view, described in "Oracle Database Vault Public Views" on page D-10.
p_value	Identity value, up to 1024 characters in mixed-case. To find the identities for each factor in the current database instance, use the DBA_DV_IDENTITY view, described in "Oracle Database Vault Public Views" on page D-10.

GET_FACTOR Function

This function is exposed to the DVF schema to allow the public factor functions to resolve the identity of a factor. This enables the F$ functions in the DVF schema. This function is available (to execute) to the general database account population.

Syntax

```
GET_FACTOR(
  p_factor VARCHAR2);
```

Parameter

Table H–3 GET_FACTOR Parameter

Parameter	Description
p_factor	Factor name. To find the available factors in the current database instance, use the DBA_DV_FACTOR view, described in "Oracle Database Vault Public Views" on page D-10.

Return Values

Returns the name of the factor in VARCHAR2 format.

GET_TRUST_LEVEL Function

This function returns the trust level of the current session identity for the factor requested. This function is available (to execute) to the general database account population. See "Creating and Configuring an Identity" on page 7-10 for a listing of the available trust levels.

Syntax

```
GET_TRUST_LEVEL(
  p_factor VARCHAR2);
```

Parameter

Table H–4 GET_TRUST_LEVEL Parameter

Parameter	Description
`p_factor`	Factor name. To find the available factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.

Return Values

Returns the trust level of the current session, in `VARCHAR2` format.

GET_TRUST_LEVEL_FOR_IDENTITY Function

This function returns the trust level for the factor and identity requested. This function is available (to execute) to the general database account population. See "Creating and Configuring an Identity" on page 7-10 for a listing of the available trust levels.

Syntax

```
GET_TRUST_LEVEL_FOR_IDENTITY(
  p_factor   VARCHAR2,
  p_identity VARCHAR2);
```

Parameters

Table H–5 GET_TRUST_LEVEL_FOR_IDENTITY Parameters

Parameter	Description
`p_factor`	Factor name. To find the available factors in the current database instance, use the `DBA_DV_FACTOR` view, described in "Oracle Database Vault Public Views" on page D-10.
`p_identity`	Identity value. To find the identities for each factor in the current database instance, use the `DBA_DV_IDENTITY` view, described in "Oracle Database Vault Public Views" on page D-10.

Return Values

Returns the trust level in `VARCHAR2` format.

ROLE_IS_ENABLED Function

This function returns a boolean value that specifies whether or not a database role has been enabled. This function is available (to execute) to the general database account population.

Syntax

```
ROLE_IS_ENABLED(
  p_role VARCHAR2);
```

Parameter

Table H–6 ROLE_IS_ENABLED Parameter

Parameter	Description
`p_role`	Database role name to check. To find existing roles, use the following views: ■ `DBA_ROLES`: Finds available roles in the current database instance. See *Oracle Database Reference*. ■ `DVA_DV_REALM_AUTH`: Finds the authorization of a particular role. See "Oracle Database Vault Public Views" on page D-10. ■ `DBA_DV_ROLE`: Finds existing secure application roles used in privilege management. See "Oracle Database Vault Public Views" on page D-10.

GET_FACTOR_LABEL Function

This 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 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. See "Label Identity" on page 7-11 for more information about factor labels.

Syntax

```
GET_FACTOR_LABEL(
  p_factor      IN VARCHAR2,
  p_policy_name IN VARCHAR2);
```

Parameters

Table H–7 GET_FACTOR_LABEL Parameters

Parameter	Description
`p_factor`	Factor name. To find the available factors in the current database instance, use the `DBA_DV_FACTOR` view. To find factors that are associated with Oracle Label Security policies, use `DBA_DV_MAC_POLICY_FACTOR`. Both are described in "Oracle Database Vault Public Views" on page D-10.

Table H–7 (Cont.) GET_FACTOR_LABEL Parameters

Parameter	Description
p_policy_name	Oracle Label Security policy name. Use the following views to find information about policies and factors in the current database instance: ■ DBA_DV_MAC_POLICY: Lists Oracle Label Security policies defined in the current database instance. ■ DBA_DV_MAC_POLICY_FACTOR: Lists the factors that are associated with Oracle Label Security policies for the current database instance. ■ DBA_DV_POLICY_LABEL: Lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy. See "Oracle Database Vault Public Views" on page D-10 for more information.

Return Values

Returns the factor label in VARCHAR2 format.

Oracle Database Vault 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. These functions are created and maintained as the Oracle Database Vault configuration API (DVSYS.DBMS_MACADM) is called for managing the various factors. The functions are then available to the general database account population through PL/SQL functions and standard SQL. This allows factors to be used in Oracle Label Security, Oracle Virtual Private Database (VPD), and so on.

For example, an account can log in to SQL*Plus and use the functions exposing the factors:

```
SQL> SELECT DVF.F$DATABASE_INSTANCE FROM dual;
F$DATABASE_INSTANCE
------------------------------------------------
ORCL
```

The name of the factor itself is case-insensitive. For example, the following statements return the same result:

```
SQL> SELECT DVF.F$database_instance FROM dual;

SQL> SELECT DVF.F$Database_Instance FROM dual;
```

Table H–8 describes the functions that are created during installation based on the default factors provided by Oracle Database Vault.

Table H–8 Installed Oracle Database Vault Factor Functions

DVF Factor Function	Description
DVF.F$AUTHENTICATION_METHOD	Returns the method of authentication. In the list that follows, the type of user is followed by the method returned: ■ Password-authenticated enterprise user, local database user, or SYSDBA/SYSOPER using Password File; proxy with user name using password: PASSWORD ■ Kerberos-authenticated enterprise or external user: KERBEROS ■ SSL-authenticated enterprise or external user: SSL ■ Radius-authenticated external user: RADIUS ■ Operating system-authenticated external user or SYSDBA/SYSOPER: OS ■ DCE-authenticated external user: DCE ■ Proxy with certificate, distinguished name (DN), or user name without using password: NONE You can use IDENTIFICATION_TYPE to distinguish between external and enterprise users when the authentication method is Password, Kerberos, or SSL.
DVF.F$CLIENT_IP	Returns the IP address and retrieval method for a client to the database server.
DVF.F$DATABASE_DOMAIN	Returns the domain of the database as specified in the DB_DOMAIN initialization parameter.
DVF.F$DATABASE_HOSTNAME	Returns the host name and retrieval method for a database.
DVF.F$DATABASE_INSTANCE	Returns the instance identifier and retrieval method for a database instance.
DVF.F$DATABASE_IP	Returns the IP address and retrieval method for a database server.
DVF.F$DATABASE_NAME	Returns the name of the database as specified in the DB_NAME initialization parameter.
DVF.F$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. 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.

Table H–8 (Cont.) Installed Oracle Database Vault Factor Functions

DVF Factor Function	Description
DVF.F$ENTERPRISE_IDENTITY	Returns the enterprise-wide identity for a user: ■ For enterprise users: the Oracle Internet Directory DN. ■ For external users: the external identity (Kerberos principal name, Radius and DCE schema names, operating system user name, certificate DN). ■ For local users and SYSDBA/SYSOPER logins: NULL. The value of the attribute differs by proxy method: ■ For a proxy with DN: the Oracle Internet Directory DN of the client. ■ For a proxy with certificate: the certificate DN of the client for external users; the Oracle Internet Directory DN for global users. ■ For a proxy with user name: the Oracle Internet Directory DN if the client is an enterprise user; NULL if the client is a local database user.
DVF.F$IDENTIFICATION_TYPE	Returns the way the schema of a user was created in the database. Specifically, it reflects the IDENTIFIED clause in the CREATE/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
DVF.F$LANG	Returns the ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter.
DVF.F$LANGUAGE	Returns the language and territory currently used by your session, along with the database character set, in the following form: *language_territory.characterset*
DVF.F$MACHINE	Returns the computer (host) name for the database client that established the database session.
DVF.F$NETWORK_PROTOCOL	Returns the network protocol being used for communication, as specified in the PROTOCOL=*protocol* portion of the connect string.
DVF.F$PROXY_ENTERPRISE_IDENTITY	Returns the Oracle Internet Directory DN when the proxy user is an enterprise user.
DVF.F$PROXYUSER	Name of the database user who opened the current session on behalf of SESSION_USER.
DVF.F$SESSION_USER	Returns the database user name by which the current user is authenticated. This value remains the same throughout the session.

Oracle Database Vault PL/SQL Rule 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 H–9 describes the functions that are created during installation based on the default rule sets provided by Oracle Database Vault.

Table H–9 Installed Oracle Database Vault PL/SQL Rule Set Functions

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

Oracle Database Vault PL/SQL Packages

Oracle Database Vault provides a collection of PL/SQL package APIs to support the maintenance and run-time behavior of Oracle Database Vault. Table H–10 lists these packages. Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package" describes these packages in detail.

Table H–10 Oracle Database Vault Administrator and Run-Time PL/SQL Packages

Package	Description
`DVSYS.DBMS_MACADM`	This package API provides for the administration of all aspects of the secure and access control configuration data. The realm owner of the Oracle Database Vault realm can grant the ability to run this package. See Appendix E, "Oracle Database Vault DVSYS.DBMS_MACADM Package" for more information.

Table H–10 (Cont.) Oracle Database Vault Administrator and Run-Time PL/SQL

Package	Description
DVSYS.DBMS_MACSEC_ROLES	This package API provides the CAN_SET_ROLE method to check 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 API also provides a method to issue the SET ROLE statement for a Oracle Database Vault Secure Application Role. 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 package is available to the general database account population. See Appendix F, "Oracle Database Vault DVSYS.DBMS_MACSEC_ROLES Package" for more information.
DVSYS.DBMS_MACUTL	This package API 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. See Appendix G, "Oracle Database Vault DVSYS.DBMS_MACUTL Package" for more information.

Note: There are several procedures in the DVSYS.DBMS_MACADM package that are not exposed in the Oracle Database Vault Administration Web application. The procedures that are not exposed include:

- CREATE_DOMAIN_IDENTITY
- CREATE_FACTOR_TYPE
- DELETE_FACTOR_TYPE
- RENAME_FACTOR_TYPE
- UPDATE_FACTOR_TYPE

I

Oracle Database Vault Security Guidelines

This appendix describes security guidelines for Oracle Database Vault. It includes the following sections:

- Accounts and Roles Trusted by Oracle Database Vault
- Accounts and Roles That Should be Limited to Trusted Individuals
- Secure Configuration Guidelines

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 I–1 lists the trusted roles and privileges that are created when you install Oracle Database Vault.

Table I–1 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 cannot add himself or herself to realm authorizations, nor can users who have the DV_ACCTMGR role alter this user.
SYSDBA privilege	Enabled	Privilege created during Oracle Database installation. Required by some Oracle features. See "Managing SYSDBA Access" on page I-2 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 "Managing SYSOPER Access" on page I-2 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.

- Managing Users with Root Access to the Operating System
- Managing the Oracle Software Owner
- Managing SYSDBA Access

- Managing SYSOPER Access

Managing Users with Root Access to the Operating System

Users who have root user access have full control over the system, including the following activities:

- 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.

Managing the Oracle Software Owner

Users who have access to a system as the Oracle software owner have control over the Oracle software, including the following activities:

- Disabling Oracle Database Vault in the given system
- 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.

Managing SYSDBA Access

Oracle Database Vault does not provide full protection against users with SYSDBA access. Several Oracle Database components require SYSDBA access. These components are:

- Oracle Data Guard and Data Guard Broker command line utilities
- Oracle Recovery Manager command line utility
- Oracle Real Application Clusters
- Oracle Automatic Storage Management command line utilities

If your installation requires SYSDBA access, Oracle recommends that you add users who need this privilege to the OSDBA group for UNIX systems and the ORA_DBA group for Windows systems. Remember that SYSDBA actions are audited by default. See *Oracle Database Installation Guide* for more information about the OSDBA group and *Oracle Database Platform Guide for Microsoft Windows* for information about ORA_DBA.

Managing SYSOPER Access

By default, Oracle Database limits SYSOPER access to operating system users in the SYSOPER group and the user SYS. It prevents SYSOPER from modifying the Oracle data dictionary directly. The SYSOPER role has limited privileges within the database, but individuals with this role can start and shut down the Oracle database. Only grant the SYSOPER role to trusted individuals.

Secure Configuration Guidelines

Follow these configuration and security guidelines:

- Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages
- Security Considerations for the Recycle Bin
- Security Considerations for the CREATE ANY JOB and CREATE JOB Privileges
- Security Considerations for the CREATE EXTERNAL JOB Privilege
- Security Considerations for the LogMiner Packages
- Security Considerations for the ALTER SYSTEM and ALTER SESSION Privileges
- Security Considerations for Java Stored Procedures and Oracle Database Vault
- Security Considerations for External C Callouts and Oracle Database Vault

Note: Be aware of the following:

- 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 in as SYSTEM and issue the following query.

  ```
  SQL> SELECT * FROM dba_tab_privs
    2  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:

  ```
  SQL> SELECT owner, name, type
    2  FROM all_dependencies
    3  WHERE referenced_name = package_name;
  ```

  Note that these two queries do not identify references to packages made through dynamic SQL.

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 in order 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.

To secure the DBMS_FILE_TRANSFER package, do the following:

- 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 and Editing a Command Rule" on page 6-4 for information on creating command rules by using Oracle Database Vault Administrator.

Alternatively, Example I–1 and Example I–2 show you can use the Oracle Database Vault MACADM package to create command rules that limit and enable access to the CREATE DATABASE LINK statement that is used to establish connections to remote databases. To use this method, log in to SQL*Plus using the Oracle Database Vault Owner account.

Example I–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;
```

When a user needs to use this command, the Oracle Database Vault owner can re-enable it from Oracle Database Vault Administrator or issue the following commands in SQL*Plus.

Example I–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;
```

Similarly, Example I–3 shows command rules that disable and enable access to CREATE DIRECTORY.

Example I–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

In this release of Oracle Database Vault, the RECYCLE BIN feature has been disabled, and a command rule has been created to prevent it from being turned on. If you need to use the RECYCLE BIN, disable the command rule ALTER SYSTEM and then enable the RECYCLE BIN as follows:

```
SQL> ALTER SYSTEM SET RECYCLEBIN=ON;
```

Security Considerations for the CREATE ANY JOB and CREATE JOB Privileges

In this release of Oracle Database Vault, the CREATE JOB privilege has been revoked from the DBA and the SCHEDULER_ADMIN roles. Ensure that this change does not affect your applications.

Security Considerations for the CREATE EXTERNAL JOB Privilege

The CREATE EXTERNAL JOB privilege was introduced in Oracle Database 10*g* Release 2 (10.2). It is required for database users who want to execute jobs that run on the operating system outside the database. By default, this 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

In this release of Oracle Database Vault, the role EXECUTE_CATALOG_ROLE no longer has EXECUTE privileges granted by default on the following LogMiner packages:

- DBMS_LOGMNR
- DBMS_LOGMNR_D
- DBMS_LOGMNR_LOGREP_DICT
- DBMS_LOGMNR_SESSION

Ensure that this change does not affect your applications.

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.

Example I–4 shows how you can create such a rule. This rule prevent users with `ALTER SYSTEM` privilege from issuing the command `ALTER SYSTEM DUMP`. Log in to SQL*Plus as the Oracle Database Vault Owner when you create this command rule.

Example I–4 Adding Rules to the Existing ALTER SYSTEM Command Rule

```
SQL> CONNECT dv_owner_acct
Enter password: password
SQL> begin
   2 dbms_macadm.create_rule('NO_SYSTEM_DUMP',
   3 '(INSTR(UPPER(DVSYS.DV_SQL_TEXT),''DUMP'') = 0)');
   4 end;
/
SQL> exec dbms_macadm.add_rule_to_rule_set
   2 ('Check trigger init parameter','NO_SYSTEM_DUMP');
SQL> 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 5-5 for more information.

Security Considerations for Java Stored Procedures and Oracle Database Vault

A definer's rights stored procedure relies on the privileges of the owner of the stored procedure to access objects referenced within the stored procedure. Invoker's rights stored procedures rely on the privileges of the executor of the stored procedure to access objects referenced within the stored procedure. The default for Java stored procedures is invoker’s rights.

Oracle Database Vault security works by intercepting calls made within the Oracle Database.

For Java stored procedures with definer’s rights, the execution of the stored procedure is not blocked and realm protection is not enforced. However, underlying objects accessed by the Java stored procedure can be protected by Oracle Database Vault command rules.

For Java stored procedures with invoker’s rights, the execution of the stored procedure is not blocked. However, underlying objects accessed by the Java stored procedure are protected by both Oracle Database Vault realms and command rules.

Limiting Access to Java Stored Procedures

By default, the `EXECUTE ANY PROCEDURE` privilege is granted to the `DBA`, `EXPORT_FULL_DATABASE`, and `IMPORT_FULL_DATABASE` roles. You can limit access to Java stored procedures by revoking the `EXECUTE ANY PROCEDURE` from users and roles who do not require it, and then by selectively assigning them read privileges. Note also that revoking the EXECUTE ANY PROCEDURE from users further secures the database by limiting access to SYS-owned packages.

Securing Java Stored Procedures

Oracle recommends that you analyze your Java stored procedures when using Oracle Database Vault to maximize security. You can do so by following these steps:

- Step 1: Identifying the Java Stored Procedures Created with Definer’s Rights
- Step 2: Finding Java Stored Procedures That Access Realm-Protected Objects
- Step 3: Creating a Package to Wrap Procedures Accessing Realm-Protected Objects
- Step 4: Identifying the Java Stored Procedures Created with Invoker’s Rights
- Step 5: Blocking Execution of Java Stored Procedures
- Step 6: Verifying Oracle Database Vault Protection for Java Stored Procedures
- Step 7: Securing Invoker’s Rights for New Java Stored Procedures

Step 1: Identifying the Java Stored Procedures Created with Definer’s Rights

Identify the Java stored procedures that were created with definers rights by running the query in Example I–5. This query returns only Java stored procedures that connect to the database, and then it spools the results to the file java_dr.lst.

Example I–5 Query to Identify Java Stored Procedures with Definers Rights

```
COLUMN plsql_owner FORMAT a8
COLUMN plsql FORMAT a30
COLUMN java_owner FORMAT a8
COLUMN java FORMAT a30
SPOOL java_dr
select distinct plu.name plsql_owner, plo.name plsql,
                ju.name java_owner, jo.name java
 from obj$ plo, user$ plu, user$ ju, obj$ jo, procedurejava$ j
 where jo.name=j.classname and ju.user#=jo.owner# and ju.name=j.ownername
   and jo.type#=29 and bitand(jo.flags, 8)=0
   and plo.owner#=plu.user#
   and j.obj#=plo.obj# and bitand(plo.flags, 8)=0
   and ju.name not in ('SYS', 'ORDSYS')
   and jo.obj# in
   (select d_obj# from dependency$ connect by d_obj#=prior p_obj#
   start with p_obj#=(select obj# from obj$ where name='java/sql/Connection'
      and owner#=0));
SPOOL off
```

Step 2: Finding Java Stored Procedures That Access Realm-Protected Objects

Analyze the Java stored procedures you queried in Step 1 and determine whether any of them access Realm protected objects. You can find a list of the realm-secured objects in the current database instance by using the DBA_DV_REALM_OBJECT view, which is described in Table D–4, " Oracle Database Vault Views" on page D-11.

Step 3: Creating a Package to Wrap Procedures Accessing Realm-Protected Objects

For Java stored procedures that do access realm-protected objects, create a PL/SQL package to wrap the Java stored procedure. Due to PL/SQL optimizations, the PL/SQL package wrapper *must* have a dummy variable defined in the package header. Adding the dummy variable enables Oracle Database Vault to intercept and block execution of Java stored procedures. Bear in mind that while this method does secure the execution of the Java stored procedure, it does not provide protection against calls to other Java stored procedures that may be embedded.

Example I–6 shows the PL/SQL package `mypackage` being created to wrap the Java class `emp_count`.

Example I–6 Creating a PL/SQL Wrapper

```
SQL> CREATE OR REPLACE PACKAGE SCOTT.MYPACKAGE AS
    tmp varchar2(200) := 'TEST';  -- dummy variable
    FUNCTION empcount RETURN VARCHAR2;
    end;
    /

Package created.

SQL> CREATE OR REPLACE PACKAGE BODY SCOTT.MYPACKAGE AS
    FUNCTION empcount RETURN VARCHAR2 AS LANGUAGE JAVA
    NAME 'emp_count.count() return java.lang.String';
    END;
    /

Package body created.
```

Step 4: Identifying the Java Stored Procedures Created with Invoker’s Rights

Next, you are ready to identify the Java stored procedures that were created with invoker’s rights. Do so by running the query in Example I–7. This query only returns Java stored procedures that connect to the database, and then it spools the results to the file `java_dr.lst`.

Example I–7 Identifying Java Stored Procedures with Invoker’s Rights

```
COLUMN plsql_owner FORMAT a8
COLUMN plsql FORMAT a30
COLUMN java_owner FORMAT a8
COLUMN java FORMAT a30
spool java_ir

select distinct plu.name plsql_owner, plo.name plsql,
                 ju.name java_owner, jo.name java
 from obj$ plo, user$ plu, user$ ju, obj$ jo, procedurejava$ j
 where jo.name=j.classname and ju.user#=jo.owner# and ju.name=j.ownername
   and jo.type#=29 and bitand(jo.flags, 8)=8
   and plo.owner#=plu.user#
   and j.obj#=plo.obj# and bitand(plo.flags, 8)=0
   and ju.name not in ('SYS', 'ORDSYS')
   and jo.obj# in
   (select d_obj# from dependency$ connect by d_obj#=prior p_obj#
   start with p_obj#=(select obj# from obj$ where name='java/sql/Connection'
      and owner#=0));

spool off
```

Step 5: Blocking Execution of Java Stored Procedures

Oracle Database Vault realm and command rules are enforced for invoker’s rights stored procedures. However, it can be useful to even block execution on Java stored procedures. You can do this by following Step 3: Creating a Package to Wrap Procedures Accessing Realm-Protected Objects.

Step 6: Verifying Oracle Database Vault Protection for Java Stored Procedures

Verify that Oracle Database Vault is protecting your Java stored procedures. Example I–8 show how you can test Oracle Database Vault security. Log in to a tool such as SQL*Plus. Then try to access a realm-protected object directly and execute a Java stored procedure to access a realm protected object.

Example I–8 Testing Oracle Database Vault Protection for Java Stored Procedures

```
SQL> connect u1
Enter password: password
Connected.
SQL> select * from session_privs;

PRIVILEGE
----------------------------------------
CREATE SESSION
SELECT ANY TABLE
CREATE PROCEDURE
EXECUTE ANY PROCEDURE

Protecting access on direct SQL access

SQL> select count(*) from scott.emp;
select count(*) from scott.emp
                           *
ERROR at line 1:
ORA-01031: insufficient privileges

Now show protecting access through Java

SQL> select scott.mypackage.empcount from dual;
select scott.mypackage.empcount from dual
       *
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SCOTT.MYPACKAGE", line 2
```

Step 7: Securing Invoker’s Rights for New Java Stored Procedures

If you are writing new Java stored procedures, ensure that Java classes execute with invoker’s rights and define them in a PL/SQL package specification. Remember, it is important to include a dummy PL/SQL variable in the package header. Adding the dummy variable enables Oracle Database Vault to intercept and block execution of Java stored procedures.

Security Considerations for External C Callouts and Oracle Database Vault

For external C callouts with definer’s rights, the execution of the callout is not blocked and realm protection is not enforced. However, underlying objects accessed by the external C callout are protected by Oracle Database Vault command rules. The default for external C callouts is invoker’s rights.

For external C callouts with invoker’s rights, the execution of the external C callout is not blocked. However, underlying objects accessed by the external C callouts are protected by both Oracle Database Vault realms and command rules.

Oracle Database Vault security works by intercepting calls made within the Oracle Database.

Securing EXECUTE ANY PROCEDURE by Limiting Access to External C Callouts

By default the `EXECUTE ANY PROCEDURE` privilege is granted to the `DBA`, `EXPORT_FULL_DATABASE`, and `IMPORT_FULL_DATABASE` roles. You can limit access to external C callouts by revoking the `EXECUTE ANY PROCEDURE` from users and roles who do not require it. Note also that revoking the `EXECUTE ANY PROCEDURE` from users further secures the database by limiting access to `SYS`-owned packages.

Securing External C Callouts

Oracle recommends that you analyze your external C callouts to maximize security when using Oracle Database Vault. You can do so by following these steps:

- Step 1: Identifying the External C Callouts Created with Definer's Rights
- Step 2: Finding the External C Callouts That Access Realm-Protected Objects
- Step 3: Creating a Package to Wrap C Callouts That Access Realm-Protected Objects
- Step 4: Identifying the External C Callouts Created with Invoker's Rights
- Step 5: Blocking Execution of Java Stored Procedures
- Step 6: Verifying Oracle Database Vault Protection for External C Callouts
- Step 7: Securing Invoker's Rights for New External C Callouts

Step 1: Identifying the External C Callouts Created with Definer's Rights

Identify the external C callouts that were created with definer's rights by running the query in Example I–9. This query spools the results to the file `external_wrap.lst`.

Example I–9 Identifying External C Callouts That Are Wrapped by PL/SQL Packages

```
spool external_wrap
select u.name OWNER, o.name object, o.type#, o.flags from
  sys.obj$ o, sys.user$ u
where o.owner# = u.user# and
u.name not in ('MDSYS', 'ORDSYS', 'SYS') and o.obj# in (
select d_obj# from dependency$ connect by d_obj#=prior p_obj#
   start with p_obj# in (select obj# from library$ where property = 0))
order by owner, object;
spool off
```

Step 2: Finding the External C Callouts That Access Realm-Protected Objects

Analyze the external C callouts and determine whether any of them access realm-protected objects. You can find a list of the realm-secured objects in the current database instance by using the `DBA_DV_REALM_OBJECT` view, which is described in Table D–4, " Oracle Database Vault Views" on page D-11.

Step 3: Creating a Package to Wrap C Callouts That Access Realm-Protected Objects

For external C callouts that do access realm-protected objects, create a PL/SQL package to wrap the external C callout. Due to PL/SQL optimizations, the PL/SQL package wrapper *must* have a dummy variable defined in the package header. Adding the dummy variable enables Oracle Database Vault to intercept and block execution of external C callout stored procedures. Bear in mind that while this method does secure the execution of the external C callout, it does not provide protection against calls to other external C callouts that may be embedded.

Example I–10 Creating a PL/SQL Wrapper

```
create or replace package scott.mytestpkg1 as
tmp integer;  /* create a dummy plsql variable */
function test return binary_integer;
end;
/

create or replace package body scott.mytestpkg1 as
function test return binary_integer as language C library
c_utils name "test" with context parameters(context,
    return indicator short,
          return int);
end;
/
```

Step 4: Identifying the External C Callouts Created with Invoker's Rights

Identify the external C callouts that were created with invoker's rights by running the query in Example I–11. This query spool the results to the file `external_standalone.lst`.

Example I–11 Identifying External C Callouts That Are Wrapped by PL/SQL Packages

```
spool external_standalone
select u.name OWNER, o.name object, o.type#, o.flags from
  sys.obj$ o, sys.user$ u
where o.owner# = u.user# and
u.name not in ('MDSYS', 'ORDSYS', 'SYS') and
o.type# in (7,8) and o.obj# in (
select d_obj# from dependency$ connect by d_obj#=prior p_obj#
   start with p_obj# in (select obj# from library$ where property = 0))
order by owner, object;
spool off
```

Step 5: Blocking Execution of Java Stored Procedures

Oracle Database Vault realm and command rules are enforced for external C callouts. However, it can be useful to even block execution on external C callouts. You can accomplish this by following Step 3: Creating a Package to Wrap C Callouts That Access Realm-Protected Objects.

Step 6: Verifying Oracle Database Vault Protection for External C Callouts

Verify Oracle Database Vault protection for external C callouts. Example I–12 shows how you can test Oracle Database Vault security by logging into a tool such as SQL*Plus and attempting to execute an external C callout.

Example I–12 Testing Oracle Database Security for an External C Callout

```
SQL> connect u1
Enter password: password
Connected.
SQL>
SQL> select * from session_privs;

PRIVILEGE
```

```
----------------------------------------
CREATE SESSION
SELECT ANY TABLE
CREATE PROCEDURE
EXECUTE ANY PROCEDURE

SQL>
SQL> select count(*) from scott.emp;
select count(*) from scott.emp
                           *
ERROR at line 1:
ORA-01031: insufficient privileges

SQL> select test from dual;
TEST
--------------------------------------------------------------------------------
14

SQL>
SQL> select scott.mypackage1.test from dual;
select scott.mypackage1.test from dual
       *
ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SCOTT.MYPACKAGE1", line 2
```

Step 7: Securing Invoker's Rights for New External C Callouts

If you are writing new external C callouts, ensure they are wrapped in an invoker's rights PL/SQL package specification. Remember, it is important to include a dummy PL/SQL variable in the package header. Adding the dummy variable enables Oracle Database Vault to intercept and block execution of external C callouts.

J

Troubleshooting Oracle Database Vault

This appendix includes the following sections:

- Using Trace Files to Diagnose Events in the Database
- General Diagnostic Tips
- Configuration Problems with Oracle Database Vault Components

Using Trace Files to Diagnose Events in the Database

You can monitor your Oracle Database Vault database instance for server and background process events by checking the database instance trace files. Trace files reveal events such as the logic that the Oracle Database Vault security enforcement engine executes, as well as internal errors, block corruption errors, deadlock errors, administrative actions that may have occurred, values of parameters that had nondefault settings when the database instance started, and other information.

Be careful about enabling trace files, however. Doing so can increase the overhead of the database instance operation, which could decrease performance. Contact Oracle Support before you decide to enable tracing.

To enable tracing, log on with any account that has the `ALTER SESSION` privilege and issue the following statement:

```
SQL> ALTER SESSION SET EVENTS '47998 trace name context forever, level 12'
```

For example, suppose you have an account that is trying to use a statement that is protected by a command rule, but the statement is not working as expected. You can diagnose the enforcement logic for this account by granting it the `ALTER SESSION` privilege, issuing the `ALTER SESSION` statement, and then retrying the statement. Afterward, check the trace files to determine what is going on.

You can disable tracing by issuing the following statement:

```
SQL> ALTER SESSION SET EVENTS '47998 trace name context off'
```

For more information about how to manage trace files, see *Oracle Database Administrator's Guide* and *Oracle Database Performance Tuning Guide*.

General Diagnostic Tips

Follow these general tips for diagnosing problems in realms, factors, and rule sets:

- 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 `EXECUTE` privileges 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 11-4 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 11-3
- "Factor Configuration Issues Report" on page 11-3
- "Factor Without Identities Report" on page 11-3
- "Identity Configuration Issues Report" on page 11-3
- "Realm Authorization Configuration Issues Report" on page 11-4
- "Rule Set Configuration Issues Report" on page 11-4
- "Secure Application Configuration Issues Report" on page 11-4

To run these reports, see "How to Run Oracle Database Vault Reports" on page 11-2.

Index