Oracle® Database Vault

Administrator's Guide

10*g* Release 2 (10.2)

B25166-20

August 2010

ORACLE®

Oracle Database Vault Administrator's Guide 10*g* Release 2 (10.2)

B25166-20

Copyright © 2006, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Patricia Huey

Contributors: Priya Badnar, Tammy Bednar, Tom Best, Ji-won Byun, Ben Chang, Martin Cheng, Chi Ching Chui, Scott Gaetjen, Viksit Gaur, Janis Greenberg, Lijie Heng, Sumit Jeloka, Dominique Jeunot, Terri Keller, Peter Knaggs, Chon Lee, Frank Lee, Paul Needham, Deborah Owens, Yi Ouyang, Hozefa Palitanawala, Robert Pang, Rupa Parameswaran, Gayathri Sairamkrishnan, Vipin Samar, Javed Samuel, James Spiller, Ashwini Supur, Srividya Tata, Kamal Tbeileh, Sudheesh Varma, Peter Wahl, Rodney Ward, Daniel Wong, Aravind Yalamanchi

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

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

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface xix

Audience xix
Documentation Accessibility xix
Related Documents xx
Conventions xx

1 Introducing Oracle Database Vault

What Is Oracle Database Vault? 1-1
Components of Oracle Database Vault 1-2
Oracle Database Vault Access Control Components 1-2
Oracle Database Vault Administrator (DVA) 1-3
Oracle Database Vault Configuration Assistant (DVCA) 1-3
Oracle Database Vault DVSYS and DVF Schemas 1-3
Oracle Database Vault PL/SQL Interfaces and Packages 1-3
Oracle Database Vault and Oracle Label Security PL/SQL APIs 1-3
Oracle Database Vault Reporting and Monitoring Tools 1-4
How Oracle Database Vault Addresses Compliance Regulations 1-4
How Oracle Database Vault Addresses Insider Threats 1-5
How Oracle Database Vault Allows for Flexible Security Policies 1-5
How Oracle Database Vault Addresses Database Consolidation Concerns 1-6

2 What to Expect After You Install Oracle Database Vault

Initialization and Password Parameter Settings That Change 2-1
How Oracle Database Vault Restricts User Authorizations 2-3
Using New Database Roles to Enforce Separation of Duties 2-3
Privileges That Are Revoked or Prevented from Existing Users and Roles 2-3
How Oracle Database Vault Affects Oracle Database Auditing 2-5
AUD$ Table Moved from SYS to the SYSTEM Schema 2-5
Modified AUDIT Statement Settings 2-5
Creating Oracle Virtual Private Database or Fine-Grained Auditing Policies 2-5

3 Getting Started with Oracle Database Vault

Starting Oracle Database Vault 3-1
Accessing Oracle Database Vault Pages from Oracle Enterprise Manager 3-1
Accessing Oracle Database Vault Pages from Database Control 3-1

Accessing Oracle Database Vault Pages from Grid Control ... 3-2
Starting Oracle Database Vault Administrator ... 3-4
Quick Start Tutorial: Securing a Schema from DBA Access ... 3-5
Step 1: Adding the SYSTEM User to the Data Dictionary Realm ... 3-6
Step 2: Log On as SYSTEM to Access the HR Schema ... 3-6
Step 3: Create a Realm ... 3-7
Step 4: Secure the EMPLOYEES Table in the HR Schema ... 3-7
Step 5: Create an Authorization for the Realm ... 3-8
Step 6: Test the Realm ... 3-9
Step 7: Run a Report ... 3-9
Step 8: Remove the Components for This Tutorial ... 3-10

4 Configuring Realms

What Are Realms? ... 4-1
Default Realms ... 4-2
Creating a Realm ... 4-2
Editing a Realm ... 4-3
Creating Realm-Secured Objects ... 4-4
Defining Realm Authorization ... 4-5
Disabling and Enabling a Realm ... 4-7
Deleting a Realm ... 4-7
How Realms Work ... 4-8
How Authorizations Work in a Realm ... 4-9
Example of How Realms Work ... 4-10
How Realms Affect Other Oracle Database Vault Components ... 4-11
Guidelines for Designing Realms ... 4-11
How Realms Affect Performance ... 4-12
Related Reports and Data Dictionary Views ... 4-13

5 Configuring Rule Sets

What Are Rule Sets? ... 5-1
Default Rule Sets ... 5-2
Creating a Rule Set ... 5-2
Configuring or Editing a Rule Set ... 5-5
Creating a Rule to Add to a Rule Set ... 5-5
Creating a New Rule ... 5-5
Adding Existing Rules to a Rule Set ... 5-7
Deleting a Rule Set ... 5-7
How Rule Sets Work ... 5-8
How Oracle Database Vault Evaluates Rules ... 5-8
Nesting Rules Within a Rule Set ... 5-8
Creating Rules to Apply to Everyone Except One User ... 5-8
Tutorial: Creating an E-Mail Alert for Security Violations ... 5-9
Step 1: Install and Configure the UTL_MAIL PL/SQL Package ... 5-9
Step 2: Create an E-Mail Security Alert PL/SQL Procedure ... 5-10
Step 3: Create a Rule Set and a Command Rule to Use the E-Mail Security Alert ... 5-11
Step 4: Test the E-Mail Security Alert ... 5-12

Step 5: Remove the Components for This Tutorial 5-13
Tutorial: Configuring Two-Person Integrity, or Dual Key Security 5-14
Step 1: Create Users for This Tutorial 5-14
Step 2: Create a Function to Check if User patch_boss Is Logged In 5-15
Step 3: Create Rules, a Rule Set, and a Command Rule to Control the Users' Access 5-16
Step 4: Test the Users' Access 5-17
Step 5: Remove the Components for This Tutorial 5-18
Guidelines for Designing Rule Sets 5-18
How Rule Sets Affect Performance 5-19
Related Reports and Data Dictionary Views 5-19

6 Configuring Command Rules

What Are Command Rules? 6-1
Default Command Rules 6-2
SQL Statements That Can Be Protected by Command Rules 6-3
Creating and Editing a Command Rule 6-4
Deleting a Command Rule 6-5
How Command Rules Work 6-5
Tutorial: Using a Command Rule to Control Table Creations by a User 6-6
Step 1: Connect as User SCOTT and Create a Table 6-6
Step 2: Connect Using the DVOWNER Role and Create a Command Rule 6-7
Step 3: Test the Command Rule 6-7
Step 4: Remove the Components for this Tutorial 6-8
Guidelines for Designing Command Rules 6-8
How Command Rules Affect Performance 6-9
Related Reports and Data Dictionary View 6-9

7 Configuring Factors

What Are Factors? 7-1
Default Factors 7-2
Creating a Factor 7-4
Editing a Factor 7-10
Adding an Identity to a Factor 7-10
About Factor Identities 7-11
Creating and Configuring a Factor Identity 7-11
Using Identity Mapping to Configure an Identity to Use Other Factors 7-13
Deleting a Factor 7-14
How Factors Work 7-15
How Factors Are Processed When a Session Is Established 7-15
How Factors Are Retrieved 7-16
How Factors Are Set 7-17
Tutorial: Preventing Ad Hoc Tool Access to the Database 7-17
Step 1: Enable the SCOTT User Account 7-18
Step 2: Create the Module Factor 7-18
Step 3: Create the Limit SQL*Plus Access Rule and Rule Set 7-19
Step 4: Create the CONNECT Command Rule 7-20

Step 5: Test the Ad Hoc Tool Access Restriction 7-21
Step 6: Remove the Components for This Tutorial 7-22
Tutorial: Restricting User Activities Based on Session Data 7-22
Step 1: Create an Administrative User 7-23
Step 2: Add Identities to the Domain Factor 7-23
Step 3: Map the Domain Factor Identities to the Client_IP Factor 7-24
Step 4: Create a Rule Set to Set the Hours and Select the Factor Identity 7-25
Step 5: Create a Command Rule That Uses the Rule Set 7-26
Step 6: Test the Factor Identity Settings 7-26
Step 7: Remove the Components for This Tutorial 7-27
Guidelines for Designing Factors 7-28
How Factors Affect Performance 7-29
Related Reports and Data Dictionary Views 7-29

8 Configuring Secure Application Roles for Oracle Database Vault

What Are Secure Application Roles in Oracle Database Vault? 8-1
Creating and Editing Secure Application Roles 8-2
Securing a Secure Application Role 8-3
Deleting a Secure Application Role 8-3
How Secure Application Roles Work 8-4
Tutorial: Granting Access with Database Vault Secure Application Roles 8-4
Step 1: Create Users for This Tutorial 8-4
Step 2: Enable the OE User Account 8-5
Step 3: Create the Rule Set and Its Rules 8-5
Step 4: Create the Database Vault Secure Application Role 8-6
Step 5: Grant the SELECT Privilege to the Secure Application Role 8-6
Step 6: Test the Database Vault Secure Application Role 8-6
Step 7: Remove the Components for This Tutorial 8-8
How Secure Application Roles Affect Performance 8-8
Related Reports and Data Dictionary View 8-9

9 Integrating Oracle Database Vault with Other Oracle Products

Using Oracle Database Vault with Oracle Enterprise Manager Grid Control 9-1
Setting the Database Vault Administrator URL in Oracle Enterprise Manager 9-1
Propagating Oracle Database Vault Policies to Other Databases 9-2
Using Enterprise Manager Grid Control Alerts for Oracle Database Vault Policies 9-4
Using Oracle Database Vault-Specific Reports in Enterprise Manager Grid Control 9-5
Integrating Oracle Database Vault with Enterprise User Security 9-5
Integrating Oracle Database Vault with Transparent Data Encryption 9-6
Attaching Factors to an Oracle Virtual Private Database 9-6
Integrating Oracle Database Vault with Oracle Label Security 9-6
How Oracle Database Vault Is Integrated with Oracle Label Security 9-6
Requirements for Using Oracle Database Vault with Oracle Label Security 9-7
Using Oracle Database Vault Factors with Oracle Label Security Policies 9-7
Tutorial: Integrating Oracle Database Vault with Oracle Label Security 9-9
Step 1: Create Users for This Tutorial 9-9
Step 2: Create the Oracle Label Security Policy 9-10

Step 3: Create Oracle Database Vault Rules to Control the OLS Authorization 9-11
Step 4: Update the ALTER SYSTEM Command Rule to Use the Rule Set 9-11
Step 5: Test the Authorizations 9-11
Step 6: Remove the Components for This Tutorial 9-12
Related Reports and Data Dictionary Views 9-13
Using Oracle Database Vault with Oracle Recovery Manager 9-14

10 Oracle Database Vault Objects

Oracle Database Vault Schemas 10-1
DVSYS Schema 10-1
DVF Schema 10-2
Oracle Database Vault Roles 10-2
About Oracle Database Vault Roles 10-2
Oracle Database Vault Owner Role, DV_OWNER 10-4
Oracle Database Vault Realm DBA Role, DV_REALM_OWNER 10-5
Oracle Database Vault Application Resource Owner Role, DV_REALM_RESOURCE 10-5
Oracle Database Vault Configuration Administrator Role, DV_ADMIN 10-6
Oracle Database Vault Account Manager Role, DV_ACCTMGR 10-6
Oracle Database Vault PUBLIC Role, DV_PUBLIC 10-7
Oracle Database Vault Security Analyst Role, DV_SECANALYST 10-8
Oracle Database Vault Accounts 10-9
Oracle Database Vault Data Dictionary Views 10-11
DBA_DV_CODE View 10-12
DBA_DV_COMMAND_RULE View 10-14
DBA_DV_FACTOR View 10-15
DBA_DV_FACTOR_LINK View 10-16
DBA_DV_FACTOR_TYPE View 10-17
DBA_DV_IDENTITY View 10-17
DBA_DV_IDENTITY_MAP View 10-18
DBA_DV_MAC_POLICY View 10-19
DBA_DV_MAC_POLICY_FACTOR View 10-19
DBA_DV_POLICY_LABEL View 10-20
DBA_DV_PUB_PRIVS View 10-20
DBA_DV_REALM View 10-21
DBA_DV_REALM_AUTH View 10-22
DBA_DV_REALM_OBJECT View 10-22
DBA_DV_ROLE View 10-23
DBA_DV_RULE View 10-23
DBA_DV_RULE_SET View 10-24
DBA_DV_RULE_SET_RULE View 10-25
DBA_DV_USER_PRIVS View 10-25
DBA_DV_USER_PRIVS_ALL View 10-26

11 Using the DVSYS.DBMS_MACADM Package

About the DVSYS.DBMS_MACADM Package 11-1
Realm Procedures Within DVSYS.DBMS_MACADM 11-1

ADD_AUTH_TO_REALM Procedure 11-2
ADD_OBJECT_TO_REALM Procedure 11-4
CREATE_REALM Procedure 11-5
DELETE_AUTH_FROM_REALM Procedure 11-6
DELETE_OBJECT_FROM_REALM Procedure 11-7
DELETE_REALM Procedure 11-8
DELETE_REALM_CASCADE Procedure 11-8
RENAME_REALM Procedure 11-9
UPDATE_REALM Procedure 11-9
UPDATE_REALM_AUTH Procedure 11-10
Rule Set Procedures Within DVSYS.DBMS_MACADM 11-12
ADD_RULE_TO_RULE_SET Procedure 11-12
CREATE_RULE Procedure 11-14
CREATE_RULE_SET Procedure 11-14
DELETE_RULE Procedure 11-16
DELETE_RULE_FROM_RULE_SET Procedure 11-17
DELETE_RULE_SET Procedure 11-18
RENAME_RULE Procedure 11-18
RENAME_RULE_SET Procedure 11-19
SYNC_RULES Procedure 11-19
UPDATE_RULE Procedure 11-19
UPDATE_RULE_SET Procedure 11-20
Command Rule Procedures Within DVSYS.DBMS_MACADM 11-22
CREATE_COMMAND_RULE Procedure 11-23
DELETE_COMMAND_RULE Procedure 11-24
UPDATE_COMMAND_RULE Procedure 11-24
Factor Procedures and Functions Within DVSYS.DBMS_MACADM 11-26
ADD_FACTOR_LINK Procedure 11-27
ADD_POLICY_FACTOR Procedure 11-28
CHANGE_IDENTITY_FACTOR Procedure 11-28
CHANGE_IDENTITY_VALUE Procedure 11-29
CREATE_DOMAIN_IDENTITY Procedure 11-30
CREATE_FACTOR Procedure 11-30
CREATE_FACTOR_TYPE Procedure 11-33
CREATE_IDENTITY Procedure 11-34
CREATE_IDENTITY_MAP Procedure 11-34
DELETE_FACTOR Procedure 11-35
DELETE_FACTOR_LINK Procedure 11-36
DELETE_FACTOR_TYPE Procedure 11-36
DELETE_IDENTITY Procedure 11-37
DELETE_IDENTITY_MAP Procedure 11-37
DROP_DOMAIN_IDENTITY Procedure 11-38
GET_INSTANCE_INFO Function 11-39
GET_SESSION_INFO Function 11-40
RENAME_FACTOR Procedure 11-40
RENAME_FACTOR_TYPE Procedure 11-41
UPDATE_FACTOR Procedure 11-41

UPDATE_FACTOR_TYPE Procedure 11-44
UPDATE_IDENTITY Procedure 11-44
Secure Application Role Procedures Within DVSYS.DBMS_MACADM 11-45
CREATE_ROLE Procedure 11-46
DELETE_ROLE Procedure 11-46
RENAME_ROLE Procedure 11-47
UPDATE_ROLE Procedure 11-47
Oracle Label Security Policy Procedures Within DVSYS.DBMS_MACADM 11-48
CREATE_MAC_POLICY Procedure 11-49
CREATE_POLICY_LABEL Procedure 11-50
DELETE_MAC_POLICY_CASCADE Procedure 11-51
DELETE_POLICY_FACTOR Procedure 11-52
DELETE_POLICY_LABEL Procedure 11-52
UPDATE_MAC_POLICY Procedure 11-53

12 Using the DVSYS.DBMS_MACSEC_ROLES Package

About the DVSYS.DBMS_MACSEC_ROLES Package 12-1
CAN_SET_ROLE Function 12-1
SET_ROLE Procedure 12-2

13 Using the DVSYS.DBMS_MACUTL Package

About the DVSYS.DBMS_MACUTL Package 13-1
DVSYS.DBMS_MACUTL Constants 13-1
DVSYS.DBMS_MACUTL Listing of Constants 13-1
Examples of Using the DVSYS.DBMS_MACUTL Constants 13-4
Procedures and Functions Within the DVSYS.DBMS_MACUTL Package 13-5
CHECK_DVSYS_DML_ALLOWED Procedure 13-6
GET_CODE_VALUE Function 13-7
GET_SECOND Function 13-8
GET_MINUTE Function 13-8
GET_HOUR Function 13-9
GET_DAY Function 13-9
GET_MONTH Function 13-10
GET_YEAR Function 13-11
IS_ALPHA Function 13-11
IS_DIGIT Function 13-12
IS_DVSYS_OWNER Function 13-12
IS_OLS_INSTALLED Function 13-13
IS_OLS_INSTALLED_VARCHAR Function 13-13
USER_HAS_OBJECT_PRIVILEGE Function 13-14
USER_HAS_ROLE Function 13-15
USER_HAS_ROLE_VARCHAR Function 13-16
USER_HAS_SYSTEM_PRIVILEGE Function 13-16

14 Using the Oracle Database Vault PL/SQL Interfaces

Oracle Database Vault Run-Time PL/SQL Procedures and Functions 14-1

SET_FACTOR Procedure 14-2
GET_FACTOR Function 14-2
GET_TRUST_LEVEL Function 14-3
GET_TRUST_LEVEL_FOR_IDENTITY Function 14-3
ROLE_IS_ENABLED Function 14-4
GET_FACTOR_LABEL Function 14-5
Oracle Database Vault PL/SQL Factor Functions 14-5
F$AUTHENTICATION_METHOD Function 14-7
F$CLIENT_IP Function 14-8
F$DATABASE_DOMAIN Function 14-8
F$DATABASE_HOSTNAME Function 14-8
F$DATABASE_INSTANCE Function 14-9
F$DATABASE_IP Function 14-9
F$DATABASE_NAME Function 14-9
F$DOMAIN Function 14-10
F$ENTERPRISE_IDENTITY Function 14-10
F$IDENTIFICATION_TYPE Function 14-11
F$LANG Function 14-11
F$LANGUAGE Function 14-12
F$MACHINE Function 14-12
F$NETWORK_PROTOCOL Function 14-13
F$PROXY_ENTERPRISE_IDENTITY Function 14-13
F$SESSION_USER Function 14-13
Oracle Database Vault PL/SQL Rule Functions 14-14
DV_SYSEVENT Function 14-14
DV_LOGIN_USER Function 14-15
DV_INSTANCE_NUM Function 14-15
DV_DATABASE_NAME Function 14-16
DV_DICT_OBJ_TYPE Function 14-16
DV_DICT_OBJ_OWNER Function 14-16
DV_DICT_OBJ_NAME Function 14-17
DV_SQL_TEXT Function 14-17
Oracle Database Vault PL/SQL Packages 14-18

15 Monitoring Oracle Database Vault

Security Violation Attempts 15-1
Database Configuration and Structural Changes 15-2
Security Policy Changes by Category 15-2
Security Policy Changes Detail 15-4

16 Oracle Database Vault Reports

Categories of Oracle Database Vault Reports 16-1
Who Can Run the Oracle Database Vault Reports? 16-1
How to Run Oracle Database Vault Reports 16-2
Generating Oracle Database Vault Reports 16-2
Oracle Database Vault Configuration Issues Reports 16-2
Command Rule Configuration Issues Report 16-3

Factor Configuration Issues Report ... 16-3
Factor Without Identities Report ... 16-3
Identity Configuration Issues Report ... 16-3
Realm Authorization Configuration Issues Report ... 16-3
Rule Set Configuration Issues Report ... 16-4
Secure Application Configuration Issues Report ... 16-4
Oracle Database Vault Auditing Reports ... 16-4
Realm Audit Report ... 16-4
Command Rule Audit Report ... 16-4
Factor Audit Report ... 16-4
Label Security Integration Audit Report ... 16-5
Core Database Vault Audit Trail Report ... 16-5
Secure Application Role Audit Report ... 16-5
Generating General Security Reports ... 16-5
Object Privilege Reports ... 16-5
Object Access By PUBLIC Report ... 16-6
Object Access Not By PUBLIC Report ... 16-6
Direct Object Privileges Report ... 16-6
Object Dependencies Report ... 16-6
Database Account System Privileges Reports ... 16-6
Direct System Privileges By Database Account Report ... 16-7
Direct and Indirect System Privileges By Database Account Report ... 16-7
Hierarchical System Privileges by Database Account Report ... 16-7
ANY System Privileges for Database Accounts Report ... 16-7
System Privileges By Privilege Report ... 16-7
Sensitive Objects Reports ... 16-7
Execute Privileges to Strong SYS Packages Report ... 16-7
Access to Sensitive Objects Report ... 16-8
Public Execute Privilege To SYS PL/SQL Procedures Report ... 16-8
Accounts with SYSDBA/SYSOPER Privilege Report ... 16-8
Privilege Management - Summary Reports ... 16-9
Privileges Distribution By Grantee Report ... 16-9
Privileges Distribution By Grantee, Owner Report ... 16-9
Privileges Distribution By Grantee, Owner, Privilege Report ... 16-9
Powerful Database Accounts and Roles Reports ... 16-9
WITH ADMIN Privilege Grants Report ... 16-10
Accounts With DBA Roles Report ... 16-10
Security Policy Exemption Report ... 16-10
BECOME USER Report ... 16-10
ALTER SYSTEM or ALTER SESSION Report ... 16-10
Password History Access Report ... 16-11
WITH GRANT Privileges Report ... 16-11
Roles/Accounts That Have a Given Role Report ... 16-11
Database Accounts With Catalog Roles Report ... 16-11
AUDIT Privileges Report ... 16-11
OS Security Vulnerability Privileges Report ... 16-11
Initialization Parameters and Profiles Reports ... 16-11

Security Related Database Parameters Report 16-11
Resource Profiles Report 16-12
System Resource Limits Report 16-12
Database Account Password Reports 16-12
Database Account Default Password Report 16-12
Database Account Status Report 16-12
Security Audit Report: Core Database Audit Report 16-12
Other Security Vulnerability Reports 16-13
Java Policy Grants Report 16-13
OS Directory Objects Report 16-13
Objects Dependent on Dynamic SQL Report 16-13
Unwrapped PL/SQL Package Bodies Report 16-13
Username/Password Tables Report 16-14
Tablespace Quotas Report 16-14
Non-Owner Object Trigger Report 16-14

A Auditing Oracle Database Vault

Oracle Database Vault Specific Audit Events A-1
Oracle Database Vault Audit Events A-1
Format of the Oracle Database Vault Audit Trail A-2
Archiving and Purging the Oracle Database Vault Audit Trail A-4
Oracle Database Audit Settings Created for Oracle Database Vault A-6

B Disabling and Enabling Oracle Database Vault

When You Must Disable Oracle Database Vault B-1
Checking if Oracle Database Vault Is Enabled or Disabled B-2
Step 1: Disable Oracle Database Vault B-2
Step 2: Perform the Required Tasks B-4
Step 3: Enable Oracle Database Vault B-5

C Oracle Database Vault Security Guidelines

Separation of Duty Guidelines C-1
How Oracle Database Vault Handles Separation of Duty C-1
Defining Separate Tasks in an Oracle Database Vault Environment C-2
Creating a Separation of Duty Matrix C-2
Identifying and Documenting the Tasks of Users Who Access the Database System C-3
Managing Oracle Database Administrative Accounts C-4
Using the SYSTEM User Account for General Administrative Uses C-4
Using the SYSTEM Schema for Application Tables C-4
Limiting the SYSDBA Privilege C-4
Managing Root and Operating System Access C-5
Accounts and Roles Trusted by Oracle Database Vault C-5
Accounts and Roles That Should be Limited to Trusted Individuals C-5
Managing Users with Root Access to the Operating System C-6
Managing the Oracle Software Owner C-6
Managing SYSDBA Access C-6

Managing SYSOPER Access C-6
Guidelines for Using Oracle Database Vault in a Production Environment C-6
Secure Configuration Guidelines C-7
Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages C-8
Security Considerations for the Recycle Bin C-9
Security Considerations for the CREATE ANY JOB and CREATE JOB Privileges C-10
Security Considerations for the CREATE EXTERNAL JOB Privilege C-10
Security Considerations for the LogMiner Packages C-10
Security Considerations for the ALTER SYSTEM and ALTER SESSION Privileges C-10
Security Considerations for Java Stored Procedures and Oracle Database Vault C-11
Limiting Access to Java Stored Procedures C-11
Securing Java Stored Procedures C-12
Step 1: Identify the Java Stored Procedures Created with Definer's Rights C-12
Step 2: Find the Java Stored Procedures That Access Realm-Protected Objects C-12
Step 3: Create a Package to Wrap Procedures Accessing Realm-Protected Objects C-12
Step 4: Identify the Java Stored Procedures Created with Invoker's Rights C-13
Step 5: Block the Execution of Java Stored Procedures C-13
Step 6: Verify Oracle Database Vault Protection for Java Stored Procedures C-13
Step 7: Secure the Invoker's Rights for New Java Stored Procedures C-14
Security Considerations for External C Callouts and Oracle Database Vault C-14
Securing EXECUTE ANY PROCEDURE by Limiting Access to External C Callouts C-14
Securing External C Callouts C-15
Step 1: Identifying the External C Callouts Created with Definer's Rights C-15
Step 2: Finding the External C Callouts That Access Realm-Protected Objects C-15
Step 3: Creating a Package to Wrap C Callouts That Access Realm-Protected Objects C-15
Step 4: Identifying the External C Callouts Created with Invoker's Rights C-16
Step 5: Blocking Execution of External C Callouts C-16
Step 6: Verifying Oracle Database Vault Protection for External C Callouts C-16
Step 7: Securing Invoker's Rights for New External C Callouts C-17

D Troubleshooting Oracle Database Vault

Using Trace Files to Diagnose Events in the Database D-1
General Diagnostic Tips D-2
Configuration Problems with Oracle Database Vault Components D-2

Index

List of Examples

4–1 Unauthorized User Trying to Create a Table 4-9
4–2 Unauthorized User Trying to Use the DELETE ANY TABLE Privilege 4-10
4–3 Authorized User Performing DELETE Operation 4-10
7–1 Using DVSYS.GET_FACTOR to Retrieve a Factor 7-16
13–1 Creating a Realm Using DVSYS.DBMS_MACUTL Constants 13-4
13–2 Creating a Rule Set Using DVSYS.DBMS_MACUTL Constants 13-5
13–3 Creating a Factor Using DVSYS.DBMS_MACUTL Constants 13-5
C–1 Creating a Command Rule to Deny Access to CREATE DATABASE LINK C-8
C–2 Creating a Command Rule to Enable Access to CREATE DATABASE LINK C-8
C–3 Command Rules to Disable and Enable Access to CREATE DIRECTORY C-9
C–4 Adding Rules to the Existing ALTER SYSTEM Command Rule C-11
C–5 Query to Identify Java Stored Procedures with Definers Rights C-12
C–6 Creating a PL/SQL Wrapper C-13
C–7 Identifying Java Stored Procedures with Invoker's Rights C-13
C–8 Testing Oracle Database Vault Protection for Java Stored Procedures C-14
C–9 Identifying External C Callouts That Are Wrapped by PL/SQL Packages C-15
C–10 Creating a PL/SQL Wrapper C-15
C–11 Identifying External C Callouts That Are Wrapped by PL/SQL Packages C-16
C–12 Testing Oracle Database Security for an External C Callout C-16

List of Figures

1–1 Oracle Database Vault Security 1-6
3–1 Oracle Database Vault Administrator Home Page 3-5
4–1 How Authorizations Work for Realms and Realm Owners 4-11
9–1 Encrypted Data and Oracle Database Vault 9-6
10–1 Hierarchy of Oracle Database Vault Roles 10-3

List of Tables

1–1 Regulations That Address Potential Security Threats 1-4
2–1 Modified Database Initialization Parameter Settings 2-1
2–2 Privileges Oracle Database Vault Revokes 2-4
2–3 Privileges Oracle Database Vault Prevents 2-4
4–1 Reports Related to Realms 4-13
4–2 Data Dictionary Views Used for Realms 4-13
5–1 Reports Related to Rule Sets 5-19
5–2 Data Dictionary Views Used for Rules and Rule Sets 5-19
6–1 Default Command Rules 6-2
6–2 Reports Related to Command Rules 6-9
7–1 Reports Related to Factors and Their Identities 7-29
7–2 Data Dictionary Views Used for Factors and Factor Identities 7-30
8–1 Reports Related to Secure Application Roles 8-9
9–1 Reports Related to Database Vault and Oracle Label Security Integration 9-13
9–2 Data Dictionary Views Used for Oracle Label Security 9-13
10–1 Privileges of Oracle Database Vault Roles 10-3
10–2 Database Accounts Used by Oracle Database Vault 10-9
10–3 Model Oracle Database Vault Database Accounts 10-10
11–1 DVSYS.DBMS_MACADM Realm Configuration Procedures 11-2
11–2 ADD_AUTH_TO_REALM Parameters 11-2
11–3 ADD_OBJECT_TO_REALM Parameters 11-4
11–4 CREATE_REALM Parameters 11-5
11–5 DELETE_AUTH_FROM_REALM Parameters 11-6
11–6 DELETE_OBJECT_FROM_REALM Parameters 11-7
11–7 DELETE_REALM Parameter 11-8
11–8 DELETE_REALM_CASCADE Parameter 11-8
11–9 RENAME_REALM Parameters 11-9
11–10 UPDATE_REALM Parameters 11-9
11–11 UPDATE_REALM_AUTH Parameters 11-11
11–12 DVSYS.DBMS_MACADM Rule Set Configuration Procedures 11-12
11–13 ADD_RULE_TO_RULE_SET Parameters 11-13
11–14 CREATE_RULE Parameters 11-14
11–15 CREATE_RULE_SET Parameters 11-15
11–16 DELETE_RULE Parameter 11-17
11–17 DELETE_RULE_FROM_RULE_SET Parameters 11-17
11–18 DELETE_RULE_SET Parameter 11-18
11–19 RENAME_RULE Parameters 11-18
11–20 RENAME_RULE_SET Parameters 11-19
11–21 UPDATE_RULE Parameters 11-20
11–22 UPDATE_RULE_SET Parameters 11-21
11–23 DVSYS.DBMS_MACADM Command Rule Configuration Procedures 11-22
11–24 CREATE_COMMAND_RULE Parameters 11-23
11–25 DELETE_COMMAND_RULE Parameters 11-24
11–26 UPDATE_COMMAND_RULE Parameters 11-25
11–27 DVSYS.DBMS_MACADM Factor Configuration Procedures 11-26
11–28 ADD_FACTOR_LINK Parameters 11-27
11–29 ADD_POLICY_FACTOR Parameters 11-28
11–30 CHANGE_IDENTITY_FACTOR Parameters 11-28
11–31 CHANGE_IDENTITY_VALUE Parameters 11-29
11–32 CREATE_DOMAIN_IDENTITY Parameters 11-30
11–33 CREATE_FACTOR Parameters 11-31
11–34 CREATE_FACTOR_TYPE Parameters 11-33
11–35 CREATE_IDENTITY Parameters 11-34

11–36 CREATE_IDENTITY_MAP Parameters 11-35
11–37 DELETE_FACTOR Parameter 11-36
11–38 DELETE_FACTOR_LINK Parameters 11-36
11–39 DELETE_FACTOR_TYPE Parameters 11-37
11–40 DELETE_IDENTITY Parameters 11-37
11–41 DELETE_IDENTITY_MAP Parameters 11-38
11–42 DROP_DOMAIN_IDENTITY Parameters 11-39
11–43 GET_INSTANCE_INFO Parameter 11-39
11–44 GET_SESSION_INFO Parameter 11-40
11–45 RENAME_FACTOR Parameters 11-40
11–46 RENAME_FACTOR_TYPE Parameters 11-41
11–47 UPDATE_FACTOR 11-41
11–48 UPDATE_FACTOR_TYPE Parameters 11-44
11–49 UPDATE_IDENTITY Parameters 11-45
11–50 DVSYS.DBMS_MACADM Secure Application Role Configuration Procedures 11-45
11–51 CREATE_ROLE Parameters 11-46
11–52 DELETE_ROLE Parameter 11-47
11–53 RENAME_ROLE Parameters 11-47
11–54 UPDATE_ROLE Parameters 11-48
11–55 DVSYS.DBMS_MACADM Oracle Label Security Configuration Procedures 11-48
11–56 CREATE_MAC_POLICY Parameters 11-49
11–57 Oracle Label Security Merge Algorithm Codes 11-49
11–58 CREATE_POLICY_LABEL Parameters 11-51
11–59 DELETE_MAC_POLICY_CASCADE Parameter 11-52
11–60 DELETE_POLICY_FACTOR Parameters 11-52
11–61 DELETE_POLICY_LABEL Parameters 11-53
11–62 UPDATE_MAC_POLICY 11-54
12–1 DVSYS.DBMS_MACSEC_ROLES Oracle Label Security Configuration Procedures 12-1
12–2 CAN_SET_ROLE Parameter 12-2
12–3 SET_ROLE Parameter 12-2
13–1 DVSYS.DBMS_MACUTL Listing of Constants 13-1
13–2 DVSYS.DBMS_MACUTL Utility Functions 13-5
13–3 CHECK_DVSYS_DML_ALLOWED Parameter 13-6
13–4 GET_CODE_VALUE Parameters 13-7
13–5 GET_SECOND Parameter 13-8
13–6 GET_MINUTE Parameter 13-8
13–7 GET_HOUR Parameter 13-9
13–8 GET_DAY Parameter 13-10
13–9 GET_MONTH Parameter 13-10
13–10 GET_YEAR Parameter 13-11
13–11 IS_ALPHA Parameter 13-11
13–12 IS_DIGIT Parameter 13-12
13–13 IS_DVSYS_OWNER Parameter 13-12
13–14 USER_HAS_OBJECT_PRIVILEGE Parameters 13-14
13–15 USER_HAS_ROLE Parameters 13-15
13–16 USER_HAS_ROLE_VARCHAR Parameters 13-16
13–17 USER_HAS_SYSTEM_PRIVILEGE Parameters 13-17
14–1 DVSYS Functions 14-1
14–2 SET_FACTOR Parameters 14-2
14–3 GET_FACTOR Parameter 14-2
14–4 GET_TRUST_LEVEL Parameter 14-3
14–5 GET_TRUST_LEVEL_FOR_IDENTITY Parameters 14-4
14–6 ROLE_IS_ENABLED Parameter 14-4
14–7 GET_FACTOR_LABEL Parameters 14-5
14–8 Installed Oracle Database Vault Factor Functions 14-6

14–9 Installed Oracle Database Vault PL/SQL Rule Set Functions 14-14
14–10 Oracle Database Vault Administrator and Run-Time PL/SQL Packages 14-18
A–1 Oracle Database Vault Audit Trail Format A-2
A–2 Audit Policy Settings Oracle Database Vault Adds to Oracle Database A-7
C–1 Example Separation of Duty Matrix C-3
C–2 Example Application Protection Matrix C-4
C–3 Trusted Oracle Database Vault Roles and Privileges C-5

Preface

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

This preface contains the following topics:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

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

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. 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.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information refer to the following documents:

- *Oracle Database Vault Release Notes*
- *Oracle Database Vault Installation Guide*
- *Oracle Label Security Administrator's Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database SQL 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/database-security/database-vault/index.html

For frequently asked questions about Oracle Database Vault, visit

http://www.oracle.com/technology/deploy/security/database-security/database-vault/dbv_faq.html

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 contains:

- 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 restricts access to specific areas in an Oracle database from any user, including users who have administrative access. For example, you can restrict administrative access to employee salaries, customer medical records, or other sensitive information.

This enables you to apply fine-grained access control to your sensitive data in a variety of ways. 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 is an integral component of your enterprise.

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

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, in multiple Oracle homes, and in Oracle Real Application Clusters (Oracle RAC) environments.

For frequently asked questions about Oracle Database Vault, visit

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

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

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

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 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 Database 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 16, "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 16, "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 *Oracle Database Vault Installation Guide*.

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.

Chapter 10, "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 Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces" and Chapter 11, "Using the 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 includes an Oracle

Policy Manager application that allows 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 9-6 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 16, "Oracle Database Vault Reports" for more information about the reports that you can generate. Chapter 15, "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
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

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

Regulation	Potential Security Threat
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 and access to the database by unauthorized applications.

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

You can associate rules and factors with 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.

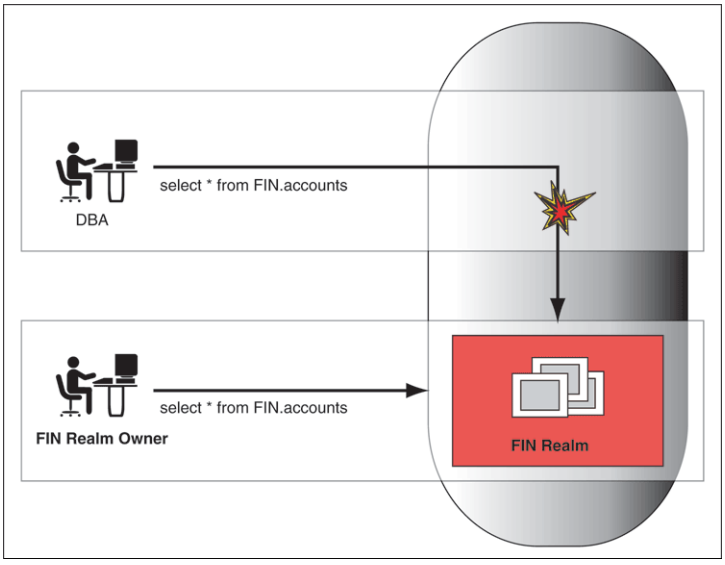
How Oracle Database Vault Addresses Database Consolidation Concerns

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 contains:

- Initialization and Password Parameter Settings That Change
- How Oracle Database Vault Restricts User Authorizations
- Using New Database Roles to Enforce Separation of Duties
- Privileges That Are Revoked or Prevented from Existing Users and Roles
- How Oracle Database Vault Affects Oracle Database Auditing
- Creating Oracle Virtual Private Database or Fine-Grained Auditing Policies

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

Initialization and Password Parameter Settings That Change

When you install Oracle Database Vault, the installation process modifies several database initialization parameter settings to better secure your database configuration. If these changes adversely affect your organizational processes or database maintenance procedures, you can revert to the original 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 top-level operations directly issued by user `SYS`, and users connecting with `SYSDBA` or `SYSOPER` privilege. For more information about `AUDIT_SYS_OPERATIONS`, see *Oracle Database Reference*.

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

Parameter	Default Value in Database	New Value Set by Database Vault	Impact of the Change
OS_AUTHENT_PREFIX	ops$	Null string	Eliminates the addition of a prefix to operating system account names. For more information about OS_AUTHENT_PREFIX, see *Oracle Database SQL 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 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 Reference*.
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*.

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

Parameter	Default Value in Database	New Value Set by Database Vault	Impact of the Change
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 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 duty*. Oracle Database Vault makes clear separation between the account management responsibility, data security responsibility, and database resource management responsibility inside the database. This means that the concept of a superprivileged user (for example, DBA) is divided among several new database roles to ensure no one user has full control over both the data and configuration of the system. Oracle Database Vault prevents the SYS user and other accounts with the DBA role and other system privileges from 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 10-2 for detailed information about the roles created during the Oracle Database Vault installation. See also "Oracle Database Vault Accounts" on page 10-9 for default accounts that are created and for suggestions of additional accounts that you may want to create.

Privileges That Are Revoked or Prevented from Existing Users and Roles

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

Table 2–2 lists privileges that Oracle Database Vault revokes from existing users and roles. Be aware that if you disable Oracle Database Vault, these privileges remain revoked. If your applications depend on these privileges, then grant them to application owner directly.

Table 2–2 Privileges Oracle Database Vault Revokes

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

Table 2–3 lists privileges that Oracle Database Vault prevents. When Oracle Database Vault is enabled, users who have the Database Vault Account Manager role (DV_ACCTMGR) have the privileges listed in this table. If you disable Oracle Database Vault, users SYS and SYSTEM have these privileges.

Table 2–3 Privileges Oracle Database Vault Prevents

User or Role	Privilege That Is Prevented
SYS user[1]	■ ALTER PROFILE ■ ALTER USER ■ CREATE PROFILE ■ CREATE USER ■ DROP PROFILE ■ DROP USER

Table 2–3 (Cont.) Privileges Oracle Database Vault Prevents

User or Role	Privilege That Is Prevented
SYSTEM user	■ ALTER PROFILE ■ ALTER USER ■ CREATE PROFILE ■ CREATE USER ■ DROP PROFILE ■ DROP USER

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

See Also:

- Table 10–1, " Privileges of Oracle Database Vault Roles" on page 10-3
- "Oracle Database Vault Account Manager Role, DV_ACCTMGR" on page 10-6

How Oracle Database Vault Affects Oracle Database Auditing

This section contains:

- AUD$ Table Moved from SYS to the SYSTEM Schema
- Modified AUDIT Statement Settings

AUD$ Table Moved from SYS to the SYSTEM Schema

In an Oracle Database Vault environment, when Oracle Label Security is enabled, the AUD$ table is moved from the SYS schema to the SYSTEM schema. The synonym SYS.AUD$ is created to refer to the SYSTEM.AUD$ table.

Tip: For greater security, create a realm around the SYSTEM.AUD$ and SYS.FGA_LOG$ tables. See Chapter 4, "Configuring Realms," for more information about realms.

Modified AUDIT Statement Settings

When you install Oracle Database Vault, it configures several AUDIT settings in the database. See Oracle Database Audit Settings Created for Oracle Database Vault for more information.

Creating Oracle Virtual Private Database or Fine-Grained Auditing Policies

If users plan to create Oracle Virtual Private Database or fine-grained auditing policies, they must have the EXECUTE privilege on the DBMS_RLS PL/SQL package. When Oracle Database Vault is enabled, the SYS user no longer owns this package; the Oracle Database Vault administrator (DV_ADMIN) does. As the DV_ADMIN user, grant these users the EXECUTE privilege for the DBMS_RLS PL/SQL package.

3

Getting Started with Oracle Database Vault

This chapter contains:

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

Starting Oracle Database Vault

This section contains:

- Accessing Oracle Database Vault Pages from Oracle Enterprise Manager
- Starting Oracle Database Vault Administrator

Accessing Oracle Database Vault Pages from Oracle Enterprise Manager

This section contains:

- Accessing Oracle Database Vault Pages from Database Control
- Accessing Oracle Database Vault Pages from Grid Control

Accessing Oracle Database Vault Pages from Database Control

From Database Control, you can use the Oracle Database Vault pages to monitor a Database Vault-enabled database and view Database Vault reports.

However, you cannot create policies or perform other Database Vault Administrator-related actions from Database Control. If you want to perform these tasks, use Database Vault Administrator, described under "Starting Oracle Database Vault Administrator" on page 3-4. If you want to propagate Database Vault policies to other Database Vault-enabled databases, then see "Accessing Oracle Database Vault Pages from Grid Control" on page 3-2.

To access the Oracle Database Vault pages from Database Control:

1. From My Oracle Support (formerly Oracle*MetaLink*), download and install patch 760748.1 (Managing Database Vault From Grid Control 10.2.0.5).

   You can access My Oracle Support from the following Web site:

   `https://support.oracle.com`

2. Start Database Control.

   For example:

   ```
   https://myserver.example.com:1158/em
   ```

3. Log in to Database Control with the following information:
   - **Username:** Enter the name of a user who has been granted the appropriate Oracle Database Vault role:
     - Creating and propagating Database Vault policies: `DV_OWNER` or `DV_ADMIN`
     - Viewing Database Vault alerts and reports: `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST`
     - Creating and managing user accounts and roles: `DV_ACCTMGR`

     The Oracle Database Vault roles do not need any additional privileges (such as `SELECT ANY DICTIONARY`) to use Database Control.

     See "About Oracle Database Vault Roles" on page 10-2 for more information.
   - **Password:** Enter your password.
   - **Connect As:** Select **Normal** from the list.
   - **Save as Preferred Credential check box:** Select this check box if you want these credentials to be automatically filled in for you the next time that this page appears.
4. In the Home page, click **Server** to display the Server subpage.
5. Under Security, select **Database Vault**.

   The Oracle Database Vault home page appears.

Accessing Oracle Database Vault Pages from Grid Control

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

However, you cannot create policies or perform other Database Vault Administrator-related actions from Grid Control. If you want to perform these functions, then use Database Vault Administrator, described under "Starting Oracle Database Vault Administrator" on page 3-4.

To access the Oracle Database Vault pages from Grid Control:

1. From My Oracle Support, download and install patch 760748.1 (Managing Database Vault From Grid Control 10.2.0.5).

   You can access My Oracle Support from the following Web site:

   `https://support.oracle.com`
2. Ensure that you have configured the Grid Control target databases that you plan to use with Database Vault.

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

   For example:

   ```
   https://myserver.example.com:1158/em
   ```
4. Log in to Oracle Enterprise Manager as user `SYSMAN` and connect as `Normal`.

5. In the Oracle Enterprise Manager Database Home page (Database Home page), select the **Targets** subpage.
6. Select the **Databases** subpage.
7. In the Databases page, select the link for the Oracle Database Vault-protected database to which you want to connect.

   The home page for the database instance appears.
8. In the Home page, click **Server** to display the Server subpage.
9. Under Security, select **Database Vault**.

   The Oracle Database Vault login page appears.
10. Enter the following information:
    - **Username:** Enter the name of a user who has been granted the appropriate Oracle Database Vault role:
      - Creating and propagating Database Vault policies: `DV_OWNER` or `DV_ADMIN`
      - Viewing Database Vault alerts and reports: `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST`
      - Creating and managing user accounts and roles: `DV_ACCTMGR`

      The Oracle Database Vault roles do not need any additional privileges (such as `SELECT ANY DICTIONARY`) to use Grid Control.

      See "About Oracle Database Vault Roles" on page 10-2 for more information.
    - **Password:** Enter your password.
    - **Connect As:** Select **Normal** from the list.
    - **Save as Preferred Credential check box:** Select this check box if you want these credentials to be automatically filled in for you the next time that this page appears.
11. Click the **Login** button.

    The Oracle Database Vault home page appears.

See Also: "Using Oracle Database Vault with Oracle Enterprise Manager Grid Control" on page 9-1

Starting Oracle Database Vault Administrator

This section describes how to start Oracle Database Vault Administrator without Oracle Enterprise Manager.

> **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:

   `http://`*`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:

   ```
   http://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`. In most cases, the host name and port number are the same as Oracle Enterprise Database Control.

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

   - **UNIX:** 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
     ```

   - **Windows:** 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
```

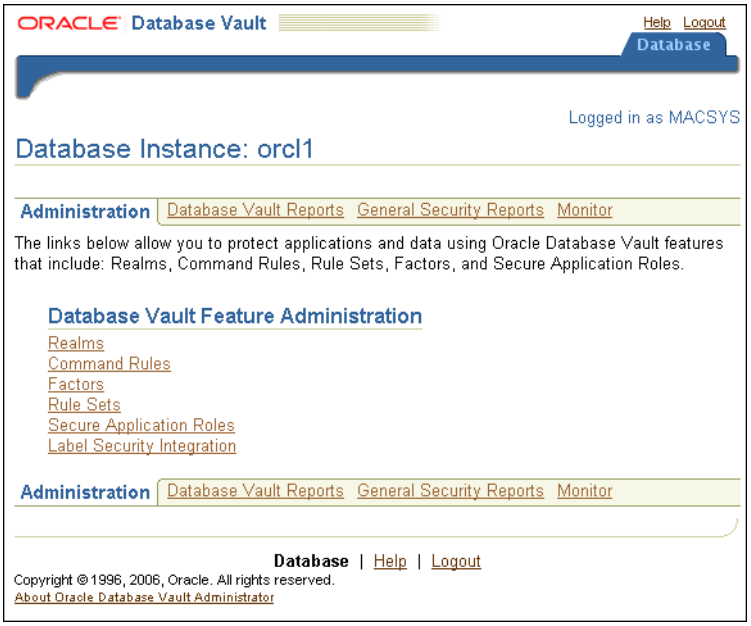
2. When the login page appears, log in by using the appropriate Oracle Database Vault account.
   - Creating and managing Database Vault policies: DV_OWNER or DV_ADMIN
   - Viewing Database Vault reports: DV_OWNER, DV_ADMIN, or DV_SECANALYST
   - Creating and managing user accounts and roles: DV_ACCTMGR

   See "About Oracle Database Vault Roles" on page 10-2 for more information.

   By default, you cannot log in to Oracle Database Vault Administrator by using the SYS, SYSTEM, or other non-Oracle Database VAult administrative accounts.

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



Quick Start Tutorial: Securing a Schema from DBA Access

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

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

In this tutorial:

- 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 Tutorial

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

In this tutorial, the `SYSTEM` user grants the `ANY` privilege to a new user account, `SEBASTIAN`. To do this, you must include `SYSTEM` in the Oracle Data Dictionary realm.

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

1. Log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Starting Oracle Database Vault" 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, 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 as user `SYSTEM` and access the `HR` schema.

```
sqlplus system
Enter password: password

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

Output similar to the following appears:

```
FIRST_NAME           LAST_NAME                     SALARY
-------------------- ------------------------- ----------
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.
```

If the HR schema is locked and expired, log in to SQL*Plus as the Database Vault Account Manager and unlock and unexpire the account. For example:

```
sqlplus amalcolm_dvacctmgr
Enter password: password

ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password
```

Replace *password* with a password that is secure.

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

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.

1. In the Realms page of Oracle Database Vault Administrator, click **Create**.
2. In the Create Realm page, under General, enter HR Realm after **Name**.
3. After Status, ensure that **Enabled** is selected so that the realm can be used.
4. Under Audit Options, ensure that **Audit On Failure** is selected so that you can create an audit trial later on.
5. 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.

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 **TABLE** from the list.

- **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 is able to view and modify the EMPLOYEES table.

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

   For example:

   ```
   CONNECT amalcolm_dvacctmgr
   Enter password: password

   CREATE USER SEBASTIAN IDENTIFIED BY password;
   ```

   Replace *password* with a password that is secure.

2. Connect as SYSTEM privilege, and then grant SEBASTIAN the following additional privileges.

   ```
   CONNECT SYSTEM
   Enter password: password

   GRANT CREATE SESSION, SELECT ANY TABLE TO SEBASTIAN;
   ```

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

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

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

1. In the Realms page of Database Vault Administrator, 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[USER]** from the list.

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

   SEBASTIAN is the only user who has access to the EMPLOYEES table in the HR schema.

4. Under Authorization Type, select **Participant**.

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

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

```
sqlplus system
Enter password: password

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

The following output should appear:

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

SYSTEM no longer has access to the salary information in the EMPLOYEES table. However, user SEBASTIAN does have access to this information. Try the following:

```
CONNECT SEBASTIAN
Enter password: password

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

Output similar to the following appears:

```
FIRST_NAME           LAST_NAME                     SALARY
-------------------- ------------------------- ----------
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.

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

   Before you can run the report, you must 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 10-2 describes these roles in detail. Currently, you should be logged in as the Database Vault Owner (DV_OWNER) account.

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 Tutorial

1. Remove the SYSTEM account from the Data Dictionary Realm.
   a. Ensure that you are logged on to Oracle Database Vault Administrator using a database account that has been granted the DV_OWNER role.
   b. From the Administration page, select **Realms**.
   c. From the list of realms, select Oracle Data Dictionary, and then click **Edit**.
   d. Under Realm Authorizations, select SYSTEM.
   e. Click **Remove**, and in the Confirmation window, click **Yes**.
2. Delete the HR Realm.
   a. In the Realms page, select HR Realm from the list of realms.
   b. 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, amalcolm_dvacctmgr) you created when you installed Oracle Database Vault, and then drop SEBASTIAN as follows:

```
sqlplus amalcolm_dvacctmgr
Enter password: password

DROP USER SEBASTIAN;
```

4. If necessary, lock and expire the HR account.

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

4

Configuring Realms

This chapter contains:

- What Are Realms?
- Default Realms
- Creating a Realm
- Editing a Realm
- Creating Realm-Secured Objects
- Defining Realm Authorization
- Disabling and Enabling a Realm
- Deleting a Realm
- How Realms Work
- How Authorizations Work in a Realm
- Example of How Realms Work
- How Realms Affect Other Oracle Database Vault Components
- Guidelines for Designing Realms
- How Realms Affect Performance
- Related Reports and Data Dictionary Views

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

prohibits any user who is not authorized to the realm to use system privileges to access the secured accounting data.

You can run reports on realms that you create in Oracle Database Vault. See "Related Reports and Data Dictionary Views" on page 4-13 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 chapters:

- Chapter 11, "Using the DVSYS.DBMS_MACADM Package"
- Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces"

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	DBSNMP	MDSYS	SYS
BI	EXFSYS	MGMT_VIEW	SYSMAN
CTXSYS	MDDATA	OUTLN	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 4-11 provides advice on creating realms.

To create a realm:

1. Log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Starting Oracle Database Vault" 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.

Oracle suggests that you use the name of the protected application as the realm name (for example, `hr_app` for an human resources application).

- **Description:** Enter a brief description of the realm. The description can contain up to 1024 characters in mixed-case. This attribute is optional.

  You may want to include a description the business objective of the given application protection and document all other security policies that compliment the realm's protection. Also document who is authorized to the realm, for what purpose, and any possible emergency authorizations.

- **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.

  For additional audit options, see "CREATE_REALM Procedure" on page 11-5.

  Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` system file, described in Appendix A, "Auditing Oracle Database Vault."

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.

     If you are creating a realm around the `AUD$` system table, then specify `SYSTEM` as the object owner. In an Oracle Database Vault environment, the `AUD$` table is moved to the `SYSTEM` schema.
   - **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 must protect, 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 encompass the entire schema specified for **Object Type** and **Object Owner**. Note that the % wildcard character applies to objects that do not yet exist and currently existing objects. Note also that the % wildcard character does not apply to roles. If you want to include multiple roles, 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 must create or access realm-secured objects
- When a user must grant or revoke realm-secured roles

A user who has been granted realm authorization as either a realm owner or a realm participant can use its system privileges to access secured objects in the realm.

Note the following:

- 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 16-3.

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.

     You cannot select yourself (that is, the user logged in) or any account that has been granted the DV_ADMIN, DV_OWNER, or DV_SECANALYST roles from this list.
   - **Authorization Type:** 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 multiple participants.
     - **Owner:** This account or role has the same privileges as the realm participant, plus the authorization to grant or revoke realm-secured database roles. The realm owner can grant privileges on realm-protected objects to other users. A realm can have multiple owners.
   - **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 occurs 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 Data Dictionary Views" on page 10-11 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 any system privilege on objects or roles protected by the realm.

   Remember that if the O7_DICTIONARY_ACCESSIBILITY initialization parameter has been set to TRUE, then non-SYS users have access to SYS schema objects. For better security, ensure that O7_DICTIONARY_ACCESSIBILITY is set to FALSE.

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 roles. Remember that a role protected by realm is not the same as an authorized 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 and 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 has full control on objects protected by the realm. Realm authorization does *not* implicitly grant extra privileges to the account. The account still must have system privileges or object privileges to access the objects. For example, an account or role may have the SELECT ANY table privilege and be a participant in the HR realm. This means the account or the account granted the role could query the HR.EMPLOYEES table. Being a participant in the realm does not mean the account or role can DROP the HR.EMPLOYEES table. Oracle Database Vault does not replace the discretionary access

control model in the existing Oracle database. It functions as a layer on top of this model for both realms and command rules.

Note the following:

- 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.
- Be aware that realm protection does not protect a table if access to the table has been granted to PUBLIC. For example, if SELECT ON *table_name* is granted to PUBLIC, then every user has access to *table_name*, even if this table is protected by a realm. As a best practice, revoke unnecessary privileges from PUBLIC.

How Authorizations Work in a Realm

Realms protect data from access through system privileges; realms do not give additional privileges to its owner or participants. The realm authorization provides a run-time 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

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

The following output should appear

```
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

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

The following output should appear:

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

Realms do not affect direct privileges on objects. For example, a user granted delete privileges to the HR.EMPLOYEES table can successfully delete records without requiring realm authorizations. Therefore, realms should minimally affect normal business application usage for database accounts.

Example 4–3 shows how an authorized user can perform standard tasks allowed within the realm.

Example 4–3 Authorized User Performing DELETE Operation

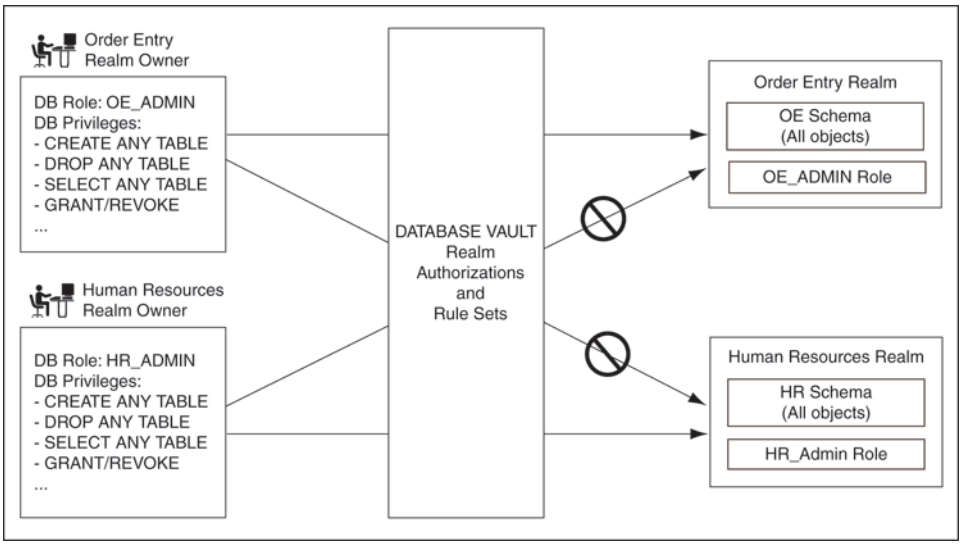
```
DELETE FROM HR.employees WHERE empno = 8002;

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 raises a realm violation, which can be audited. The powers of each realm owner are limited within the realm itself. For example, OE_ADMIN has no access to the Human Resources realm, and HR_ADMIN has no access to the Order Entry realm.

Figure 4–1 How Authorizations Work for Realms and Realm Owners



See Also: "Quick Start Tutorial: Securing a Schema from DBA Access" on page 3-5 for a tutorial on how to create and use a realm

How Realms Affect Other Oracle Database Vault Components

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

"How Realms Work" on page 4-8 explains the steps that Oracle Database Vault takes to process SQL statements that affect objects in a realm. "How Command Rules Work" on page 6-5 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 multiple realms and an account or role can be authorized in multiple realms.

  To provide limited access to a subset of a database schema (for example, just the `EMPLOYEES` table in the `HR` schema, or roles protected by a realm), create a new realm with just the minimum required objects and authorizations.

- If you want to add a role to a realm as a grantee, create a realm to protect the role. Doing so prevents users who have been granted the GRANT ANY ROLE system privilege, such as the SYSTEM user account, from granting the role to themselves.
- If you want to add the SYS user account to a realm authorization, you must add user SYS explicitly and not through a role (such as the DBA role).
- Be mindful of the privileges currently allowed to a role that you plan to add as a realm authorization.

  Realm authorization of a role can be accidentally granted and not readily apparent if an account such as SYS or SYSTEM creates a role for the first time and the Oracle Database Vault administrator adds this role as a realm authorization. This is because the account that creates a role is implicitly granted the role when it is created.
- Sometimes you must temporarily relax realm protections for an administrative task. Rather than disabling the realm, have the Security Manager (DV_ADMIN or DV_OWNER) log in, add the named account to the authorized accounts for the realm, and set the authorization rule set to Enabled. Then in the enabled rule set, turn on all auditing for the rule set. You can remove the realm authorization when the administrative task is complete.
- If you want to grant ANY privileges to new users, Oracle recommends that you add a database administrative user to the data dictionary realm so that this user can grant other users ANY privileges, if they need them. For example, using a named account to perform the GRANT of the ANY operations enables you to audit these operations, which creates an audit trail for accountability.
- 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 Database Vault 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 to perform auditing.
  3. For an Oracle Data Pump import, grant the BECOME USER privilege to the user performing the import for the duration of the Data Pump operation. If the schema accounts being imported do not exist in the target database, then grant the DV_ACCTMGR role to SYSTEM for the time frame that the import will occur. When the import is complete, you can revoke the grants from both users.

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, and also have a realm around the entire schema to protect the entire application. Be aware, however, that this type of configuration may slow performance, but it does enable you to grant realm authorization to some of the objects in a schema.

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

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 and Data Dictionary Views

Table 4–1 lists Oracle Database Vault reports that are useful for analyzing realms. See Chapter 16, "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 16-4	Audits records generated by the realm protection and realm authorization operations
"Realm Authorization Configuration Issues Report" on page 16-3	Lists authorization configuration information, such as incomplete or disabled rule sets, or nonexistent grantees or owners that may affect the realm
"Rule Set Configuration Issues Report" on page 16-4	Lists rule sets that do not have rules defined or enabled, which may affect the realms that use them
"Object Privilege Reports" on page 16-5	Lists object privileges that the realm affects
"Privilege Management - Summary Reports" on page 16-9	Provides information about grantees and owners for a realm
"Sensitive Objects Reports" on page 16-7	Lists objects that the command rule affects

Table 4–2 lists data dictionary views that provide information about existing realms.

Table 4–2 Data Dictionary Views Used for Realms

Data Dictionary View	Description
"DBA_DV_REALM View" on page 10-21	Lists the realms created in the current database instance.
"DBA_DV_REALM_AUTH View" on page 10-22	lists the authorization of a named database user account or database role (`GRANTEE`) to access realm objects in a particular realm
"DBA_DV_REALM_OBJECT View" on page 10-22	Lists the database schemas, or subsets of schemas with specific database objects contained therein, that are secured by the realms

5

Configuring Rule Sets

This chapter contains:

- 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
- Tutorial: Creating an E-Mail Alert for Security Violations
- Tutorial: Configuring Two-Person Integrity, or Dual Key Security
- Guidelines for Designing Rule Sets
- How Rule Sets Affect Performance
- Related Reports and Data Dictionary Views

What Are Rule Sets?

A **rule set** is a collection of one or more rules that you can associate with a realm authorization, factor assignment, command rule, or secure application role. The rule set evaluates to true or false based on the evaluation of each rule it contains and the evaluation type (*All True* or *Any True*). A rule within a rule set is a PL/SQL expression that evaluates to true or false. You can create a rule and add the rule to multiple rule sets.

You can use rule sets to accomplish the following activities:

- As a further restriction to realm authorization, to define the conditions under which realm authorization is active
- To define when to allow a command rule
- To enable a secure application role
- To define when to assign the identity of a factor

When you create a rule set, Oracle Database Vault makes it available for selection when you configure the authorization for a realm, command rule, factor, or secure application role.

You can run reports on the rule sets that you create in Oracle Database Vault. See "Related Reports and Data Dictionary Views" on page 5-19 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 chapters:

- Chapter 11, "Using the DVSYS.DBMS_MACADM Package"
- Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces"

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.

See also the following sections:

- "Guidelines for Designing Rule Sets" on page 5-18 for advice on designing rule sets
- "Oracle Database Vault PL/SQL Rule Functions" on page 14-14 for a set of functions that you can use in rule expressions
- "Rule Set Configuration Issues Report" on page 16-4 to check the configuration of the rule sets for your database

To create a rule set:

1. Log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

"Starting Oracle Database Vault" 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.

  Oracle suggests that you start the name with a noun and complete it with the realm or command rule name to which the rule set is attached. For example:

  ```
  Limit SQL*Plus access
  ```

- **Description:** Enter a description of the functionality for the rule set. It can have up to 1024 characters in mixed-case. This attribute is optional.

  You may want to document the business requirement of the rule set. For example:

  ```
  Rule to limit access to SQL*Plus
  ```

- **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 multiple rules 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 generate an audit trail for the rule set. Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` system file, described in Appendix A, "Auditing Oracle Database Vault."

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.

For additional rule audit options, see "CREATE_RULE_SET Procedure" on page 11-14.

The Oracle Database Vault audit trail, written to the `DVSYS.AUDIT_TRAIL$` system table, 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, "Auditing Oracle Database Vault," for more information.

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 Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces".

  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 `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 privileged user.

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.

     Oracle suggests that you start the name with a verb and complete the name with the purpose of the rule. For example:

     ```
     Prevent non-admin access to SQL*Plus
     ```

     Because rules do not have a **Description** field, make the name explicit but be sure to not exceed over 90 characters.
   - **Rule Expression:** Enter a PL/SQL expression that fits the following requirements:
     - It is valid in a SQL `WHERE` clause.
     - It can be a freestanding and valid PL/SQL Boolean expression such as the following:

       ```
       TO_CHAR(SYSDATE,'HH24') = '12'
       ```

     - It must evaluate to a Boolean (`TRUE` or `FALSE`) value.
     - It must be no more than 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 `EXECUTE` privilege on the function. Doing so reduces the chances of errors when you add new rules.
     - Ensure that the rule works. You can test the syntax by running the following statement in SQL*Plus:

       ```
       SELECT rule_expression FROM DUAL;
       ```

       For example, suppose you have created the following the rule expression:

       ```
       SYS_CONTEXT('USERENV','SESSION_USER') != 'TSMITH'
       ```

       You could test this expression as follows:

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

     See the following sections for functions that you can use in the rule set expression:
     - "Oracle Database Vault PL/SQL Rule Functions" on page 14-14
     - Chapter 11, "Using the DVSYS.DBMS_MACADM Package"
     - Chapter 13, "Using the 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 Data Dictionary Views" on page 10-11 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 Procedures Within DVSYS.DBMS_MACADM" on page 11-12. For example, to delete the rule Night Shift, log in to SQL*Plus as the Database Vault Owner and enter the following statement:

```
EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Night Shift');
```

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 Data Dictionary Views" on page 10-11 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
- 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.

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')  '22' AND '23'
```

You can create it using a factor function. See "Oracle Database Vault PL/SQL Factor Functions" on page 14-5 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, a privileged user). 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 a privileged user, then the system evaluates the rule to true without evaluating *additional_rule*. If the current user is not a privileged user, then the evaluation of the rule depends on the evaluation of *additional_rule*.

Tutorial: Creating an E-Mail Alert for Security Violations

In the following tutorial, you will create an e-mail alert that is sent when a user attempts to alter a table outside a maintenance period. To do this, you must create a rule to set the maintenance period hours, attach this rule to a rule set, and then create a command rule to allow the user to alter the table. You then associate the rule set with this command rule, which then sends the e-mail alert when the user attempts to use the `ALTER TABLE` SQL statement outside the maintenance period.

In this tutorial:

- 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 a Rule Set and a Command Rule to Use the E-Mail Security Alert
- Step 4: Test the E-Mail Security Alert
- Step 5: Remove the Components for This Tutorial

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

1. Log on to SQL*Plus as `SYS` using the `SYSDBA` privilege, and then install the `UTL_MAIL` package.

   ```
   sqlplus sys as sysdba
   Enter password: password

   @?/rdbms/admin/utlmail.sql
   @?/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 about `UTL_MAIL`.

2. Check the current value of the `SMTP_OUT_SERVER` parameter, and make a note of this value so that you can restore it when you complete this tutorial.

   For example:

   ```
   SHOW PARAMETER SMTP_OUT_SERVER

   NAME                 TYPE             VALUE
   -------------------- ---------------- ---------------------------------
   SMTP_OUT_SERVER      string           some_value.example.com
   ```

3. Issue the following `ALTER SYSTEM` statement:

   ```
   ALTER SYSTEM SET SMTP_OUT_SERVER="imap_mail_server.example.com";
   ```

   Replace *`imap_mail_server`* with the name of your SMTP server, which you can find in the account settings in your e-mail tool. Enclose these settings in quotation marks. For example:

   ```
   ALTER SYSTEM SET SMTP_OUT_SERVER="my_imap_mail_server.example.com"
   ```

4. Connect as `SYS` using the `SYSOPER` privilege and then restart the database.

   ```
   CONNECT SYS/AS SYSOPER
   Enter password: password

   SHUTDOWN IMMEDIATE
   STARTUP
   ```

5. Ensure that the SMTP_OUT_SERVER parameter setting is correct.

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

SHOW PARAMETER SMTP_OUT_SERVER

NAME                 TYPE             VALUE
-------------------- ---------------- --------------------------------
SMTP_OUT_SERVER      string           my_imap_mail_server.example.com
```

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

1. Ensure that you are connected as SYS using the SYSDBA privilege, and then grant the following privileges to a user who has been granted the DV_OWNER or DV_ADMIN role.

   For example:

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

GRANT CREATE PROCEDURE, DROP ANY PROCEDURE TO lbrown_dvowner;
GRANT EXECUTE ON UTL_TCP TO lbrown_dvowner;
GRANT EXECUTE ON UTL_SMTP TO lbrown_dvowner;
GRANT EXECUTE ON UTL_MAIL TO lbrown_dvowner;
```

   The UTL_TCP, UTL_SMTP, and UTL_MAIL PL/SQL packages are used by the e-mail security alert that you will create.

2. Connect as the Oracle Database Vault DV_OWNER or DV_ADMIN user.

   For example:

```
CONNECT lbrown_dvowner
Enter password: password
```

3. Create the following procedure:

```
CREATE OR REPLACE PROCEDURE email_alert AS
msg varchar2(20000) := 'Realm violation occurred for the ALTER TABLE Command
Security Policy rule set. The time is: ';
BEGIN
  msg := msg||to_char(SYSDATE, 'Day DD MON, YYYY HH24:MI:SS');
UTL_MAIL.SEND (
    sender      => 'youremail@example.com',
    recipients  => 'recipientemail@example.com',
    subject     => 'Table modification attempted outside maintenance!',
    message     => msg);
END email_alert;
/
```

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

4. Grant the EXECUTE permission on this procedure to DVSYS.

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

5. Test the settings that you have created so far.

```
EXEC EMAIL_ALERT;
```

```
COMMIT;
```

SQL*Plus should display a `PL/SQL procedure successfully completed` message, and in a moment, depending on the speed of your mail server, you should receive the e-mail alert.

Step 3: Create a Rule Set and a Command Rule to Use the E-Mail Security Alert

1. As the Oracle Database Vault user, create the following rule set:

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'ALTER TABLE Command Security Policy',
  description      => 'This rule set allows ALTER TABLE only during the
                       maintenance period.',
  enabled          => 'Y',
  eval_options     => 1,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
  fail_options     => 2,
  fail_message     => '',
  fail_code        => NULL,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler          => 'dbavowner.email_alert');
 END;
/
```

2. Create a rule similar to the following.

   For now, set the rule expression to be during the time you will test it. For example, if you want to test it between 2 p.m. and 3 p.m., create the rule as follows:

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Restrict Access to Maintenance Period',
  rule_expr  => 'TO_CHAR(SYSDATE,''HH24'') BETWEEN ''14'' AND ''15''');
 END;
/
```

   Ensure that you use two single quotation marks instead of double quotation marks for `HH24`, `14`, and `15`. You can double-check the system time on your computer by issuing the following SQL statement:

```
SELECT TO_CHAR(SYSDATE,'HH24') FROM DUAL;

TO
--
14
```

   Later on, when you are satisfied that the rule works, you can update it to a time when your site typically performs maintenance work (for example, between 7 p.m. and 10 p.m) as follows:

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_RULE(
  rule_name  => 'Restrict Access to Maintenance Period',
  rule_expr  => 'TO_CHAR(SYSDATE,''HH24'') BETWEEN ''19'' AND ''22''');
END;
/
```

3. Add the Restrict Access to Maintenance Period rule to the ALTER TABLE Command Security Policy rule set.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'ALTER TABLE Command Security Policy',
  rule_name     => 'Restrict Access to Maintenance Period');
 END;
 /
```

4. Create the following command rule:

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_COMMAND_RULE(
   command         => 'ALTER TABLE',
   rule_set_name   => 'ALTER TABLE Command Security Policy',
   object_owner    => 'SCOTT',
   object_name     => '%',
   enabled         => 'Y');
 END;
/
```

5. Commit these updates to the database.

```
COMMIT;
```

Step 4: Test the E-Mail Security Alert

1. Connect to SQL*Plus as user `SCOTT`.

   For example:

   ```
   CONNECT SCOTT
   Enter password: password
   ```

   If the `SCOTT` account is locked and expired, then a user with the `DV_ACCTMGR` role can unlock this account and create a new password as follows:

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

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

2. As the user `SCOTT`, create a test table.

   ```
   CREATE TABLE mytest (col1 number);
   ```

3. Change the system time on your computer to a time when the ALTER TABLE Command Security Policy rule set takes place (for example, between 2 p.m. and 3 p.m).

   UNIX: Log in as root and use the `date` command to set the time. For example, assuming the date today is December 13, 2007, you would enter the following:

   ```
   su root
   Password: password

   date 12131409
   ```

   Windows: Double-click the clock icon, which is typically at the lower right corner of the screen. In the Date and Time Properties window, set the time to 2 p.m., and then click **OK**.

4. Try altering the `my_test` table.

   ```
   ALTER TABLE mytest ADD (col2 number);
   ```

```
Table altered.
```

SCOTT should be able to alter the mytest table during this time.

5. Reset the system time to a time outside the Restrict Access to Maintenance Period time.
6. Connect as SCOTT and try altering the my_test table again.

```
CONNECT SCOTT
Enter password: password

ALTER TABLE mytest ADD (col3 number);

ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-47400: Command Rule violation for alter table on SCOTT.MYTEST
ORA-06512: at "DVSYS.AUTHORIZE_EVENT", line 55
ORA-06512: at line 31
```

SCOTT cannot alter the mytest table. In a moment, you should receive an e-mail with the subject header `Table modification attempted outside maintenance!` and with a message similar to the following:

```
Realm violation occurred for the ALTER TABLE Command Security Policy rule set.
The time is: Wednesday 28 NOV, 2007 14:24:25
```

7. Reset the system time to the correct time.

Step 5: Remove the Components for This Tutorial

1. Connect to SQL*Plus as the Oracle Database Vault DV_OWNER or DV_ADMIN user account.

   For example:

```
CONNECT lbrown_dvowner
Enter password: password
```

2. Enter the following commands in the order shown to delete the rule set components.

```
EXEC DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('ALTER TABLE Command Security
Policy', 'Restrict Access to Maintenance Period');

EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Restrict Access to Maintenance Period');

EXEC DVSYS.DBMS_MACADM.DELETE_COMMAND_RULE('ALTER TABLE', 'SCOTT', '%');

EXEC DVSYS.DBMS_MACADM.DELETE_RULE_SET('ALTER TABLE Command Security Policy');
```

3. Drop the email_alert PL/SQL procedure.

```
DROP PROCEDURE email_alert;
```

4. Connect as user SCOTT and remove the mytest table.

```
CONNECT SCOTT
Enter password: password

DROP TABLE mytest;
```

5. Connect as SYS using the SYSDBA privilege and then revoke the EXECUTE privilege on the UTL_TCP, UTL_SMTP, and UTL_MAIL PL/SQL packages from the Oracle Database Vault Owner account.

   For example:

   ```
   REVOKE EXECUTE ON UTL_TCP FROM lbrown_dvowner;
   REVOKE EXECUTE ON UTL_SMTP FROM lbrown_dvowner;
   REVOKE EXECUTE ON UTL_MAIL FROM lbrown_dvowner;
   ```

6. Set the SMTP_OUT_SERVER parameter to its original value.

   For example:

   ```
   ALTER SYSTEM SET SMTP_OUT_SERVER="some_value.my_company.com";
   ```

7. Restart the database. For example:

   ```
   CONNECT SYS/AS SYSOPER
   Enter password: password

   SHUTDOWN IMMEDIATE
   STARTUP
   ```

Tutorial: Configuring Two-Person Integrity, or Dual Key Security

In this tutorial, you configure a rule set that defines two-person integrity (TPI), also called dual key security, dual key connection, and two-man rule security. In this type of security, two users are required to authorize an action, instead of one user. The idea is that one user provides a safety check for the other user before that user can proceed with a task. Two-person integrity provides an additional layer of security for actions that can potentially be dangerous. This type of scenario is often used for tasks such as database patch updates, which is what this tutorial will demonstrate. One user, patch_user must log in to perform a database patch upgrade, but the only way that he can do this is if his manager, patch_boss is already logged in. You will create a function, rules, a rule set, and a command rule to control patch_user's ability to log in.

This section contains:

- Step 1: Create Users for This Tutorial
- Step 2: Create a Function to Check if User patch_boss Is Logged In
- Step 3: Create Rules, a Rule Set, and a Command Rule to Control the Users' Access
- Step 4: Test the Users' Access
- Step 5: Remove the Components for This Tutorial

Step 1: Create Users for This Tutorial

You must create the following two users for this tutorial:

- patch_boss acts in a supervisory role: If patch_boss is not logged in, then the patch_user user cannot log in.
- patch_user is the user who is assigned to perform the patch upgrade. However, for this tutorial, user patch_user does not actually perform a patch upgrade. He only attempts to log in.

To create the users:

1. Connect to SQL*Plus as a user who has been granted the DV_ACCTMGR role.

   For example:

   ```
   CONNECT amalcolm_dvacctmgr
   Enter password: password
   ```

2. Create the following users.

   ```
   CREATE USER PATCH_BOSS IDENTIFIED BY password;
   CREATE USER PATCH_USER IDENTIFIED BY password;
   ```

3. Connect as user SYS with the SYSDBA privilege and grant privileges to the following users.

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

   GRANT CREATE PROCEDURE TO lbrown_dvowner;
   GRANT SELECT ON V_$SESSION TO lbrown_dvowner;
   GRANT CREATE SESSION TO PATCH_BOSS;
   GRANT CREATE SESSION TO PATCH_USER;
   ```

Step 2: Create a Function to Check if User patch_boss Is Logged In

The function that you must create, check_boss_logged_in, does just that: When user patch_user tries to log in to SQL*Plus, it checks that user patch_boss is already logged in by checking the V$SESSION data dictionary view.

To create the check_boss_logged_in function:

1. Connect to SQL*Plus as a user who has been granted the DV_OWNER role.

   For example:

   ```
   CONNECT lbrown_dvowner
   Enter password: password
   ```

2. Create the check_boss_logged_in function as follows:

   ```
   CREATE OR REPLACE FUNCTION check_boss_logged_in
   return varchar2
   authid current_user as

   v_session_number number := 0;
   v_allow varchar2(10)    := 'TRUE';
   v_deny varchar2(10)     := 'FALSE';

   BEGIN
     SELECT COUNT(*) INTO v_session_number
     FROM SYS.V_$SESSION
     WHERE USERNAME = 'PATCH_BOSS'; -- Enter the user name in capital letters.

    IF v_session_number > 0
     THEN RETURN v_allow;
    ELSE
     RETURN v_deny;
    END IF;
   END check_boss_logged_in;
   /
   ```

3. Grant the EXECUTE privilege on the check_boss_logged_in function to the DVSYS schema.

```
GRANT EXECUTE ON check_boss_logged_in to DVSYS;
```

Step 3: Create Rules, a Rule Set, and a Command Rule to Control the Users' Access

Next, you must create two rules, a rule set to which you will add them, and a command rule. The rule set triggers the check_boss_logged_in function when user patch_user tries to logs in to the database.

To create the rules and rule set:

1. Create the Check if Boss Is Logged In rule, which checks that the patch_user user is logged in to the database.

   If the check_boss_logged_in function returns TRUE (that is, patch_boss is logged in to another session), then patch_user can log in.

```
BEGIN
  DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check if Boss Is Logged In',
  rule_expr => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''PATCH_USER'' and
LBROWN_DVOWNER.check_boss_logged_in =  ''TRUE'' ');
END;
/
```

   Enter the user name, PATCH_USER, in upper-case letters, which is how the SESSION_USER parameter stores it.

2. Create the Allow Connect for Other Database Users rule, which ensures that the user logged in (patch_user) is not user patch_boss. It also enables all other valid users to log in.

```
BEGIN
     DVSYS.DBMS_MACADM.CREATE_RULE(
     rule_name => 'Allow Connect for Other Database Users',
     rule_expr => 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') != ''PATCH_USER''
');
END;
/
COMMIT;
```

3. Create the Dual Connect for Boss and Patch rule set, and then add the two rules to it.

```
BEGIN
     DVSYS.DBMS_MACADM.CREATE_RULE_SET(
     rule_set_name => 'Dual Connect for Boss and Patch',
     description => 'Checks if both boss and patch users are logged in.',
     enabled           => 'Y',
     eval_options      => 2,
     audit_options     => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
     fail_options      => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
     fail_message      =>'',
     fail_code         => NULL,
     handler_options   => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
     handler           => ''
     );
END;
/
```

```
BEGIN
     DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET(
     rule_set_name     => 'Dual Connect for Boss and Patch',
     rule_name         => 'Check if Boss Is Logged In'
   );
END;
/

BEGIN
     DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET(
     rule_set_name     => 'Dual Connect for Boss and Patch',
     rule_name         => 'Allow Connect for Other Database Users'
   );
END;
/
```

4. Create the following CONNECT command rule, which permits user `patch_user` to connect to the database only if `patch_boss` is already logged in.

```
BEGIN
   DVSYS.DBMS_MACADM.CREATE_COMMAND_RULE(
   command             => 'CONNECT',
    rule_set_name      => 'Dual Connect for Boss and Patch',
    object_owner       => '%',
    object_name        => '%',
    enabled            => 'Y');
END;
/
COMMIT;
```

Step 4: Test the Users' Access

1. Try to connect to SQL*Plus as user `patch_user`.

```
CONNECT patch_user
Enter password: password

ERROR:
ORA-47400: Command Rule violation for CONNECT on LOGON
```

   User `patch_user` cannot log in until user `patch_boss` is already logged in.

2. Open a second shell and then log in as user `patch_boss`.

```
sqlplus patch_boss
Enter password: password
Connected.
```

   As you can see, user `patch_boss` can log in.

3. Go back to the first shell, and then try logging in as user `patch_user` again.

```
sqlplus patch_user
Enter password: password
Connected.
```

   This time, user `patch_user` is deemed a valid user, so now he can log in.

Step 5: Remove the Components for This Tutorial

1. In the session for the user `patch_boss`, exit SQL*Plus and then close the shell.

```
EXIT
```

2. In the first shell, connect the `DV_ACCTMGR` user and remove the users you created.

```
CONNECT amalcolm_dvacctmgr
Enter password: password

DROP USER patch_boss;
DROP USER patch_user;
```

3. Connect as the `DV_OWNER` user and delete the `check_boss_logged_in` function, the rules, rule set, and command rule.

```
CONNECT lbrown_dvowner
Enter password: password

DROP FUNCTION check_boss_logged_in;

EXEC DVSYS.DBMS_MACADM.DELETE_COMMAND_RULE('CONNECT', '%', '%');

EXEC DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('Dual Connect for Boss and
Patch', 'Check if Boss Is Logged In');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('Dual Connect for Boss and
Patch', 'Allow Connect for Other Database Users');

EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Check if Boss Is Logged In');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Allow Connect for Other Database Users');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE_SET('Dual Connect for Boss and Patch');

COMMIT;
```

4. Connect as user `SYS` with the `SYSDBA` privilege and revoke the following privileges from the `DV_OWNER` user.

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

REVOKE CREATE PROCEDURE FROM lbrown_dvowner;
REVOKE SELECT ON V_$SESSION FROM lbrown_dvowner;
```

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 "Tutorial: Creating an E-Mail Alert for Security Violations" 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 and Data Dictionary Views

Table 5–1 lists Oracle Database Vault reports that are useful for analyzing rule sets and the rules within them. See Chapter 16, "Oracle Database Vault Reports" for information about how to run these reports.

Table 5–1 Reports Related to Rule Sets

Report	Description
"Rule Set Configuration Issues Report" on page 16-4	Lists rule sets that have no rules defined or enabled
"Secure Application Configuration Issues Report" on page 16-4	Lists secure application roles that have incomplete or disabled rule sets
"Command Rule Configuration Issues Report" on page 16-3	Lists rule sets that are incomplete or disabled

Table 5–2 lists data dictionary views that provide information about existing rules and rule sets.

Table 5–2 Data Dictionary Views Used for Rules and Rule Sets

Data Dictionary View	Description
"DBA_DV_RULE View" on page 10-23	Lists the rules that have been defined

Table 5–2 (Cont.) Data Dictionary Views Used for Rules and Rule Sets

Data Dictionary View	Description
"DBA_DV_RULE_SET View" on page 10-24	Lists the rules sets that have been created
"DBA_DV_RULE_SET_RULE View" on page 10-25	Lists rules that are associated with existing rule sets

6

Configuring Command Rules

This chapter contains:

- What Are Command Rules?
- Default Command Rules
- SQL Statements That Can Be Protected by Command Rules
- Creating and Editing a Command Rule
- Deleting a Command Rule
- How Command Rules Work
- Tutorial: Using a Command Rule to Control Table Creations by a User
- Guidelines for Designing Command Rules
- How Command Rules Affect Performance
- Related Reports and Data Dictionary View

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 is enforced at run time. Command rules affect anyone who tries to use the SQL statements it protects, regardless of the realm in which the object exists. If you want to protect realm-specific objects, see "Defining Realm Authorization" on page 4-5.

A command rule has the following attributes, in addition to associating a command rule to a command:

- SQL statement the command rule protects
- Owner of the object the command rule affects
- Database object the command rule affects
- Whether the command rule is enabled
- An associated rule set

For more information about SQL statements and operations, refer to *Oracle Database SQL 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 that uses factors for the `CONNECT` event to permit or deny sessions after the usual steps–user authentication process, factor initialization, and Oracle Label Security integration–are complete.

For example, you can configure a command rule that allows DDL statements such as `CREATE TABLE`, `DROP TABLE`, and `ALTER TABLE` in the `BIZAPP` schema to be authorized after business hours, but not during business hours.

You can run reports on the command rules that you create in Oracle Database Vault. See "Related Reports and Data Dictionary View" on page 6-9 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 chapters:

- Chapter 11, "Using the DVSYS.DBMS_MACADM Package"
- Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces"

Default Command 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
`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	CREATE CONTEXT	DROP FUNCTION
ALTER DIMENSION	CREATE DATABASE LINK	DROP INDEX
ALTER FUNCTION	CREATE DIMENSION	DROP INDEXTYPE
ALTER INDEX	CREATE DIRECTORY	DROP JAVA
ALTER INDEXTYPE	CREATE FUNCTION	DROP LIBRARY
ALTER JAVA	CREATE INDEX	DROP OPERATOR
ALTER OPERATOR	CREATE INDEXTYPE	DROP OUTLINE
ALTER OUTLINE	CREATE JAVA	DROP PACKAGE
ALTER PACKAGE	CREATE LIBRARY	DROP PACKAGE BODY
ALTER PACKAGE BODY	CREATE OPERATOR	DROP PROCEDURE
ALTER PROCEDURE	CREATE OUTLINE	DROP PROFILE
ALTER PROFILE	CREATE PACKAGE	DROP ROLE
ALTER RESOURCE COST	CREATE PACKAGE BODY	DROP ROLLBACK SEGMENT
ALTER ROLE	CREATE PROCEDURE	DROP SEQUENCE
ALTER ROLLBACK SEGMENT	CREATE PROFILE	DROP SNAPSHOT
ALTER SEQUENCE	CREATE ROLE	DROP SNAPSHOT LOG
ALTER SNAPSHOT	CREATE ROLLBACK SEGMENT	DROP SYNONYM
ALTER SNAPSHOT LOG	CREATE SCHEMA	DROP TABLE
ALTER SYNONYM	CREATE SEQUENCE	DROP TABLESPACE

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

Creating and Editing a Command Rule

Follow these steps:

1. Log in to Oracle Database Vault Administrator as a user who has been granted the DV_OWNER or DV_ADMIN role.

   "Starting Oracle Database Vault" 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 affects. You can use wildcard character `%` to select all owners. (However, you cannot use wildcard characters with text, such as `EM%` to select all owners whose names begin in `EM`.) This attribute is mandatory for all SQL statements that operate on objects within a specific schema. See "SQL Statements That Can Be Protected by Command Rules" on page 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 affects, 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 and Data Dictionary View" on page 6-9 for more information.

Rule Set

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

If the rule set evaluates to true, then the SQL statement succeeds. If it evaluates to false, the statement fails, and then Oracle Database Vault raises a command rule violation. (You can track such rule violations by using the Command Rule Configuration Issues Report, discussed in Chapter 16.) 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 Data Dictionary Views" on page 10-11 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 4-8 describes what happens when a database account issues a `SELECT`, DDL, or DML statement that affects objects within a realm.

The following actions take place when `SELECT`, DDL, or DML statement is issued:

1. Oracle Database Vault queries all the command rules that need to be applied.

For SELECT, DDL, and DML statements, multiple command rules may apply because the object owner and object name support wildcard notation.

You can associate rule sets with both command rules and realm authorizations. Oracle Database Vault evaluates the realm authorization rule set first, and then it evaluates the rule sets that apply to the command type being evaluated.

2. For each command rule that applies, Oracle Database Vault evaluates its associated rule set.
3. If the associated rule set of any of the applicable command rules returns false or errors, Oracle Database Vault prevents the command from executing. Otherwise, the command is authorized for further processing. The configuration of the rule set with respect to auditing and event handlers dictates the auditing or custom processing that occurs.

   Command rules override object privileges. You can disable (or set the **Disabled** setting for) either a command or a rule set of a command. If a command rule is disabled, then the command rule is not allowed to run regardless of its associated rule set. That is, the command rule itself is disabled. However, if you disable a rule set of a command rule, then the rule set is not checked. In that case, the command rule is allowed to run without any condition. If the command rule 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 rule itself.)

Tutorial: Using a Command Rule to Control Table Creations by a User

In this tutorial, you create a simple command rule that disables and then enables the ability of user SCOTT to create tables.

In this tutorial:

- Step 1: Connect as User SCOTT and Create a Table
- Step 2: Connect Using the DVOWNER Role and Create a Command Rule
- Step 3: Test the Command Rule
- Step 4: Remove the Components for this Tutorial

See Also: "Tutorial: Creating an E-Mail Alert for Security Violations" on page 5-9 for another example of how a command rule can work with a rule set to send an e-mail alert when a violation occurs

Step 1: Connect as User SCOTT and Create a Table

1. Log in to SQL*Plus as user SCOTT.

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

If the SCOTT account is locked and expired, then log in as the Database Vault Account Manager and unlock SCOTT and create a new password. For example:

```
sqlplus amalcolm_dvacctmgr
Enter password: password

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

Replace *password* with a password that is secure.

```
CONNECT SCOTT
Enter password: password
```

2. As user SCOTT, create a table.

```
CREATE TABLE t1 (num NUMBER);
```

3. Now drop the table.

```
DROP TABLE t1;
```

At this stage, user SCOTT can create and drop tables. Do not exit SQL*Plus yet, and remain connected as SCOTT. You must use it later on when SCOTT tries to create another table.

Step 2: Connect Using the DVOWNER Role and Create a Command Rule

1. Log in to Oracle Database Vault Administrator as a user who has been granted the DV_OWNER or DV_ADMIN role.

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

2. In the Oracle Database Vault Administrator Administration page, click **Command Rules**.

   The Command Rules page appears.

3. Click **Create**.

   The Create Command Rule page appears.

4. Enter the following settings:
   - **Command list:** Select **CREATE TABLE**
   - **Status:** Set to **Enabled** so that the command rule is active.
   - **Object Owner:** Select **SCOTT**.
   - **Object Name:** Set to **%** so that it applies to all objects in the SCOTT schema.
   - **Rule Set:** Select **Disabled** so that user SCOTT is prevented from creating tables.

5. Click **OK**.

   Do not exit Database Vault Administrator

Command rules take effect immediately. Right away, user SCOTT is prevented from creating tables, even though he is still in the same user session he was in a moment ago, before you created the CREATE TABLE command rule.

Step 3: Test the Command Rule

1. In SQL*Plus, ensure that you are logged on as user SCOTT.

```
CONNECT SCOTT
Enter password: password
```

2. Try to create a table.

```
CREATE TABLE t1 (num NUMBER);
```

   The following output should appear:

```
ERROR at line 1:
```

```
ORA-00604: error occurred at recursive SQL level 1
ORA-47400: Command Rule violation for create table on SCOTT.T1
ORA-06512: at "DVSYS.AUTHORIZE_EVENT", line 55
ORA-06512: at line 31
```

As you can see, SCOTT is no longer allowed to create tables, even in his own schema.

3. In Oracle Database Vault Administrator, do the following:
   a. In the Command Rules page, select the CREATE TABLE command rule and then click **Edit**.
   b. In the Edit Command Rule page, select **Enabled** from the **Rule Set** list.
   c. Click **OK**.
4. In SQL*Plus, as user SCOTT, try creating the table again.

```
CREATE TABLE t1 (num NUMBER);

Table created.
```

Now that the CREATE TABLE command rule is set to Enabled, user SCOTT is once again permitted to create tables. (Do not exit SQL*Plus.)

Step 4: Remove the Components for this Tutorial

1. In Oracle Database Vault Administrator, remove the CREATE TABLE command rule as follows:
   - Return to the Command Rules page.
   - Select the CREATE TABLE command rule and then click **Remove**.
   - In the Confirmation page, click **Yes**.
2. Log in to SQL*Plus as user SCOTT and remove the t1 table.

```
SQL> DROP TABLE t1;
```

3. If you no longer need the SCOTT account to be available, then connect as the Database Vault Account Manager and enter the following ALTER USER statement:

```
CONNECT amalcolm_dvacctmgr
Enter password: password

ALTER USER SCOTT ACCOUNT LOCK PASSWORD EXPIRE;
```

Guidelines for Designing Command Rules

Follow these guidelines to design command rules:

- Create finer-grained command rules, because they are far easier to maintain.

  For example, if you want to prevent SELECT statements from occurring on specific schema objects, then design multiple command rules to stop the SELECT statements on those specific schema objects, rather than creating a general command rule to prevent SELECT statements in the schema level.
- When designing rules for the CONNECT event, be careful to include logic that does not inadvertently lock out the Oracle Database Vault Owner or Administrator.

If the account has been locked out, you can disable Oracle Database Vault, correct the rule that is causing the lock-out problem, and then reenable Oracle Database Vault. Even when Oracle Database Vault is disabled, you still can use Database Vault Administrator and the Database Vault PL/SQL packages. See Appendix B, "Disabling and Enabling Oracle Database Vault" for instructions on disabling and re-enabling Database Vault.

- Sometimes you must temporarily relax an enabled command rule for an administrative task. Rather than disabling the command rule, have the Security Manager (the account with the `DV_ADMIN` or `DV_OWNER` role) log in, set the rule set to **Enabled**, turn on **Auditing on Success or Failure** for the default rule set named Enabled, and then set the command rule back to its original rule set when the task is complete.
- When designing command rules, be careful to consider automated processes such as backup where these procedures may be inadvertently disabled. You can account for these tasks by creating rules that allow the command when a series of Oracle Database Vault factors is known to be true, for example, the program being used, and the account being used or the computer or network on which the client program is running.

How Command Rules Affect Performance

The performance of a command rule depends on the complexity of the rules in the rule set associated with the command rule. For example, suppose a rule set invokes a PL/SQL function that takes 5 seconds to run. In this case, a command rule that uses that rule set would take 5 seconds to grant access for the command statement to run.

You can check the system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager 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 and Data Dictionary View

Table 6–2 lists Oracle Database Vault reports that are useful for analyzing command rules. See Chapter 16, "Oracle Database Vault Reports" for information about how to run these reports.

Table 6–2 Reports Related to Command Rules

Report	Description
"Command Rule Audit Report" on page 16-4	Lists audit records generated by command rule processing operations
"Command Rule Configuration Issues Report" on page 16-3	Tracks rule violations, in addition to other configuration issues the command rule may have
"Object Privilege Reports" on page 16-5	Lists object privileges that the command rule affects
"Sensitive Objects Reports" on page 16-7	Lists objects that the command rule affects
"Rule Set Configuration Issues Report" on page 16-4	Lists rules sets that have no rules defined or enabled, which may affect the command rules that use them

You can use the DBA_DV_COMMAND_RULE data dictionary view to find the SQL statements that are protected by command rules. See "DBA_DV_COMMAND_RULE View" on page 10-14 for more information.

7

Configuring Factors

This chapter contains:

- What Are Factors?
- Default Factors
- Creating a Factor
- Editing a Factor
- Adding an Identity to a Factor
- Deleting a Factor
- How Factors Work
- Tutorial: Preventing Ad Hoc Tool Access to the Database
- Tutorial: Restricting User Activities Based on Session Data
- Guidelines for Designing Factors
- How Factors Affect Performance
- Related Reports and Data Dictionary Views

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 14-5 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 9-6 explains how. See "Tutorial:

Integrating Oracle Database Vault with Oracle Label Security" on page 9-9 for more information.

You can run reports on the factors that you create in Oracle Database Vault. See "Related Reports and Data Dictionary Views" on page 7-29 for more information.

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

- Chapter 11, "Using the DVSYS.DBMS_MACADM Package"
- Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces"

Default Factors

Oracle Database Vault provides a set of default factors. For each of these factors, there is an associated function that retrieves the value of the factor. See "Oracle Database Vault PL/SQL Factor Functions" on page 14-5 for a listing of these functions.

You can create custom factors by using your own PL/SQL retrieval methods. A useful PL/SQL function you can use (which is used for many of the default factors) is the `SYS_CONTEXT` SQL function, which retrieves data about the user session. After you create the custom factor, you can query its values similar to the functions used to query the default factors. "Tutorial: Preventing Ad Hoc Tool Access to the Database" on page 7-17 shows an example of how to create and query a custom factor.

See *Oracle Database SQL Reference* for more information about the `SYS_CONTEXT` function.

You can use the default factors in your own security configurations. If you do not need them, you can remove them. (That is, they are not needed for internal use by Oracle Database Vault.)

The default factors are as follows:

- **Authentication_Method:** Is the method of authentication. In the list that follows, the type of user is followed by the method returned:
  - Password-authenticated enterprise user, local database user, or `SYSDBA/SYSOPER` using Password File; proxy with user name using password: `PASSWORD`
  - Kerberos-authenticated enterprise or external user: `KERBEROS`
  - SSL-authenticated enterprise or external user: `SSL`
  - Radius-authenticated external user: `RADIUS`
  - Operating system-authenticated external user or `SYSDBA/SYSOPER`: `OS`
  - DCE-authenticated external user: `DCE`
  - Proxy with certificate, distinguished name (DN), or user name without using password: `NONE`

  You can use `IDENTIFICATION_TYPE` to distinguish between external and enterprise users when the authentication method is Password, Kerberos, or SSL.
- **Client_IP:** Is the IP address of the machine from which the client is connected.
- **Database_Domain:** Is the domain of the database as specified in the `DB_DOMAIN` initialization parameter.

- **Database_Hostname:** Is the host name of the computer on which the instance is running.
- **Database_Instance:** Is the instance identification number of the current instance.
- **Database_IP:** Is the IP address of the computer on which the instance is running.
- **Database_Name:** Is the name of the database as specified in the `DB_NAME` initialization parameter.
- **Domain:** Is a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level. You can identify a domain using factors such as host name, IP address, and database instance names of the Database Vault nodes in a secure access path to the database. Each domain can be uniquely determined using a combination of the factor identifiers that identify the domain. You can use these identifying factors and possibly additional factors to define the Maximum Security Label within the domain. This restricts data access and commands, depending on the physical factors about the Database Vault session. Example domains of interest may be Corporate Sensitive, Internal Public, Partners, and Customers.
- **Enterprise_Identity:** Is the enterprise-wide identity for the user:
  - For enterprise users: the Oracle Internet Directory-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` and `SYSOPER` logins: NULL.

  The value of the attribute differs by proxy method:
  - For a proxy with DN: the Oracle Internet Directory DN of the client.
  - For a proxy with certificate: the certificate DN of the client for external users; the Oracle Internet Directory DN for global users.
  - For a proxy with user names: the Oracle Internet Directory DN if the client is an enterprise user; NULL if the client is a local database user.
- **Identification_Type:** Is the way the user schema was created in the database. Specifically, it reflects the `IDENTIFIED` clause in the `CREATE/ALTER USER` syntax. In the list that follows, the syntax used during schema creation is followed by the identification type returned:
  - `IDENTIFIED BY` *`password`*: `LOCAL`
  - `IDENTIFIED EXTERNALLY`: `EXTERNAL`
  - `IDENTIFIED GLOBALLY`: `GLOBAL SHARED`
  - `IDENTIFIED GLOBALLY AS DN`: `GLOBAL PRIVATE`
- **Lang:** Is the ISO abbreviation for the language name, a shorter form than the existing `LANGUAGE` parameter.
- **Language:** Is the language and territory your session currently uses, along with the database character set, in the following form:

  `language_territory.characterset`

  For example:

  `AMERICAN_AMERICA.WE8MSWIN1252`

Refer to *Oracle Database Globalization Support Guide* for more information about languages, territories, and character sets.

- **Machine:** Is the host name for the database client that established the current session. If you must find out whether the computer was used for a client or server session, then you can compare this setting with the Database_Hostname factor to make the determination.
- **Network_Protocol:** Is the network protocol being used for communication, as specified in the `PROTOCOL=protocol` portion of the connect string.
- **Proxy_Enterprise_Identity:** Is the Oracle Internet Directory DN when the proxy user is an enterprise user.
- **Proxy_User:** Is the name of the database user who opened the current session on behalf of `SESSION_USER`.
- **Session_User:** Is the database user name by which the current user is authenticated. This value remains the same throughout the session.

Creating a Factor

In general, to create a factor, you first create the factor itself, and then you edit the factor to include its identity. "Guidelines for Designing Factors" on page 7-28 provides advice on designing factors.

To create a factor:

1. Log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Starting Oracle Database Vault" 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 creates a valid Oracle identifier for the factor function to be created in the `DVF` schema based on the name of the factor chosen. For example, if you

create a factor named GetNetworkIP, Oracle Database Vault creates the DVF.F$GETNETWORKIP function. This attribute is mandatory.

Oracle suggests that you start the name with a noun and complete the name with a brief description of the derived value.

"Oracle Database Vault PL/SQL Factor Functions" on page 14-5 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. You can create user-defined factor types, such as application name, certificate information, and so on in addition to the installed factor types, such as time and authentication method.

  You can find the factors that are associated with a particular factor type by querying the DBA_DV_FACTOR data dictionary view. For example:

```
SELECT NAME
FROM DVSYS.DBA_DV_FACTOR
WHERE FACTOR_TYPE_NAME='Authentication Method';
```

  The output is:

```
NAME
------------------------------
Network_Protocol
Authentication_Method
Identification_Type
```

  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 Procedure" on page 11-33.

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 December 6, 2008, the **By Method** option would return the following value:

```
2008-12-06
```

- **By Constant:** Resolves the factor identity by retrieving the constant value found in the **Retrieval Method** field.

- **By Factors:** Determines the factor identity by mapping the identities of the child factor to its parent factor. A parent factor is a factor whose values are resolved based on a second factor, called a child factor. To establish their relationship, you map their identities. (You do not need to specify a **Retrieval Method** expression for this option.)

  See "Using Identity Mapping to Configure an Identity to Use Other Factors" on page 7-13 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 and Data Dictionary Views" on page 7-29 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, in SQL*Plus:

```
CONNECT lbrown_dvowner
Enter password: password

SELECT DVF.F$DATABASE_IP FROM DUAL;
```

Output similar to the following appears:

```
SELECT DVF.F$DATABASE_IP FROM DUAL;

F$DATABASE_IP
-------------------------------------------------------------
192.0.2.1
```

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

```
SELECT GET_FACTOR('DATABASE_IP') FROM DUAL;
```

The following output appears:

```
GET_FACTOR('DATABASE_IP')
------------------------------------------------------------
192.0.2.1
```

Evaluation

Under Evaluation, select how you want the factor to be evaluated and assigned an identity. See "How Factors Affect Performance" on page 7-29 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) and 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 algorithm. This attribute is mandatory if you want to use an OLS label. (See also "Integrating Oracle Database Vault with Oracle Label Security" on page 9-6 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 14-5
- "Factor Procedures and Functions Within DVSYS.DBMS_MACADM" on page 11-26
- Chapter 13, "Using the 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 14-5
- "Factor Procedures and Functions Within DVSYS.DBMS_MACADM" on page 11-26
- Chapter 13, "Using the 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:

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

Then you can create an assignment rule for the GEO_STATE factor to allow or disallow the setting of the GEO_STATE factor based on other factors or rule expressions. See "How Factors Are Set" on page 7-17 for more information.

Audit Options

Under Audit Options, select from the settings to generate an audit trail for the factors. Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` system file, described in Appendix A, "Auditing Oracle Database Vault."

You can use the Factor Audit Report to display the generated audit records. (See "Related Reports and Data Dictionary Views" on page 7-29 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 a Factor Identity" on page 7-11 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 7-10
- "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 optionally 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 192.0.2.4.

This section contains:

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

About Factor Identities

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 Also:

- "How Factors Work" on page 7-15 for more information about how a factor behaves during a database session
- "Tutorial: Restricting User Activities Based on Session Data" on page 7-22 for an example of how to create and use factor identities

Creating and Configuring a Factor 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.

For example, suppose you have created a factor named Network. You can create the following identities for the Network factor:

- Intranet, with a trust level of 10
- VPN (virtual private network), with a trust level of 5
- Public, with a trust level of 1

You then can create rule expressions (or custom application code) that base policy decisions on the trust level. For example, you can use `DVSYS.GET_TRUST_LEVEL` to find trust levels greater than 5:

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

Or, you can use a `SELECT` statement on the `DVSYS.DBA_DV_IDENTITY` data dictionary view to find trust levels for the Network factor greater than or equal to 5:

```
SELECT VALUE, TRUST_LEVEL FROM DVSYS.DBA_DV_IDENTITY
   WHERE TRUST_LEVEL >= 5
   AND FACTOR_NAME='Network'
```

Output similar to the following appears:

```
F$NETWORK GET_TRUST_LEVEL('NETWORK')
-----------------------------------
VPN                                5
INTRANET                          10
```

In the preceding example, Network factor identity for VPN is trusted (value equals 5), and the identity for the INTRANET domain is 10, which implies a greater trust.

See Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces" 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 Data Dictionary Views" on page 10-11 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**.

Using Identity Mapping to Configure an Identity to Use Other Factors

After you create, edit, and save the factor 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 16-3. See also "Tutorial: Restricting User Activities Based on Session Data" on page 7-22 for an example of how to use identity mapping.

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 a Factor Identity" on page 7-11 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** list.

   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 192.0.2.1 and the **High Value** is set to 192.0.2.24. This means that whenever the client IP address lies in the specified address range of 192.0.2.1 to 192.0.2.24, the parent factor evaluates to a predefined identity (for example, INTRANET).

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 192.0.2.1 to 192.0.2.24. The REMOTE identity can map to an IP address range that excludes the address range 192.0.2.1 to 192.0.2.24.

   Based on identity mapping, you can create a security policy. For example, you can define a reduced set of privileges for an employee connecting over VPN (with REMOTE), as opposed to an employee connecting from within the corporate network (with INTRANET).

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 192.0.2.1 and 192.0.2.24. So, if an authorized accounting financial application program, running on a client with IP address 192.0.2.12 accesses the database, then the Network factor is resolved to ACCOUNTING-SENSITIVE. A database session with the ACCOUNTING-SENSITIVE Network value would have more access privileges than one with the INTRANET Network value.

Deleting a Factor

Before you delete a factor, you can locate the various references to the factor and its identities by querying the factor-related Oracle Database Vault views. See "Oracle Database Vault Data Dictionary Views" on page 10-11 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 is undefined (`NULL`).

5. Depending on the outcome of this factor evaluation, factor validation, and trust level resolution, Database Vault audits the details of the evaluation as dictated by the factor audit configuration.

6. When the evaluation of all factors that are identified by method or constant completes, Oracle Database Vault resolves the factors that are identified by other factors by using the identity maps that are defined for the factor configured identities.

   The evaluation order of the factor-configured identities is by ASCII sort on the identity values: Oracle Database Vault uses the first alphabetically sorted identity mapping that it evaluates. For example, suppose factor TEST has identities X and Y. Furthermore, identities X and Y have identity maps that are dependent on identities for factors A, B, and C. The following mapping occurs:

   - X is mapped when A=1 and B=1
   - Y is mapped when A=1, B=1, and C=2

   In this case, the first one evaluated is X. Y is not evaluated, but what if its C mapping meets the criteria that is needed for the TEST factor's success? You would need to reverse the mapping, that is, map Y before X so that A, B, and C can be evaluated first. To reverse the mapping, rename Y to V (or some alphabetic value that sorts before X) so that it can be correctly resolved.

This algorithm works if the ASCII sort ordering is correct and the identities map the same number factors at some level.

7. When the factor initialization completes, the Oracle Database Vault integration with Oracle Label Security occurs.

After this process completes, Oracle Database Vault checks to see if a command rule is associated with the CONNECT event. If a rule set associated with the CONNECT event, then Oracle Database Vault evaluates the rule set. If the rule set evaluates to false or results in an error, then the session is terminated. Oracle Database Vault executes any auditing or call handlers associated with the rule set before the session is terminated.

Note: Be careful about associating command rules with the CONNECT event, because you can inadvertently lock out other users from of the database. In general, if you create a command rule for CONNECT, set its evaluation option of the associated rule set to Any True.

If you do inadvertently lock out users, then you should temporarily disable Oracle Database Vault, disable the CONNECT command rule, re-enable Oracle Database Vault, and then fix the factor code that is causing the problem. "If the Test Fails" on page 7-21 provides an example of how to accomplish this.

How Factors Are Retrieved

You can retrieve a factor in a database session at any time by using the DVF factor function or the DVSYS.GET_FACTOR function. To find a listing of available factors, query the DVS.DBA_DV_FACTOR data dictionary view, described in"DBA_DV_FACTOR View" on page 10-15.

Example 7–1 shows an example of using the DVSYS.GET_FACTOR function.

Example 7–1 Using DVSYS.GET_FACTOR to Retrieve a Factor

```
SELECT GET_FACTOR('client_ip') FROM DUAL;
```

You can use the factor values retrieved from the DVF factor function or the DVSYS.GET_FACTOR in the following ways:

- Oracle Database Vault rule expressions
- Custom application code that is available to all database sessions in an Oracle Database Vault environment

"Oracle Database Vault PL/SQL Factor Functions" on page 14-5 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 7-15.

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 7-15, 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 7-15 occur.

If you have not associated a assignment rule set for the factor or if the rule set returns false (or returns errors), then Oracle Database Vault sends an error message if you attempt to set the factor using the `DVSYS.SET_FACTOR` function.

Tutorial: Preventing Ad Hoc Tool Access to the Database

Many database applications contain features to explicitly control the actions of a user. However, an ad hoc query tool, such as SQL*Plus, may not have these controls. As a result, a user could use an ad hoc tool to perform actions in the database that he or she would normally be prevented from performing in a regular database application. You can use a combination of Oracle Database Vault factors, rule sets, and command rules to prevent unauthorized access to the database by ad hoc query tools.

In the following tutorial, you limit the use of SQL*Plus to only four users: the Database Vault Owner, the Database Vault Account Manager, `SYSTEM`, and `SYS`. To accomplish this, you must create a factor to find the applications on your system and a rule and rule set to limit SQL*Plus to these four users. Then you create a command rule for the `CONNECT` SQL statement, which is associated with the rule set. When you successfully complete this tutorial, then only the administrative users you specify should be able to connect to the database using SQL*Plus.

In this tutorial:

- Step 1: Enable the SCOTT User Account
- Step 2: Create the Module Factor
- Step 3: Create the Limit SQL*Plus Access Rule and Rule Set
- Step 4: Create the CONNECT Command Rule
- Step 5: Test the Ad Hoc Tool Access Restriction
- Step 6: Remove the Components for This Tutorial

See Also:

- "Tutorial: Restricting User Activities Based on Session Data" on page 7-22 for an example of using factor identity mapping
- "Tutorial: Integrating Oracle Database Vault with Oracle Label Security" on page 9-9 for an example of integrating an Oracle Database Vault factor with an Oracle Label Security label

Step 1: Enable the SCOTT User Account

You must use the SCOTT account later on when you test the Oracle Database Vault components for this tutorial, so ensure that this account is active.

1. Log in to SQL*Plus as a user who has been granted the DV_ACCTMGR role.

   For example:

   ```
   sqlplus amalcolm_dvacctmgr
   Enter password: password
   ```

2. Check the status of the SCOTT account.

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

3. If the SCOTT account is expired and locked, then enter the following statement to make it active:

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

   Replace *password* with a password that is secure.

Step 2: Create the Module Factor

The Module factor uses the SYS_CONTEXT SQL function to find the names of the applications that are used to access the current instance of Oracle Database. As described previously, the SYS_CONTEXT SQL function provides many useful methods for finding the state of a user session. SYS_CONTEXT is a valuable tool for creating custom factors.

See *Oracle Database SQL Reference* for more information about the SYS_CONTEXT function.

1. Log in to Oracle Database Vault Administrator as a user who has been granted the DV_OWNER or DV_ADMIN role.

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

2. In the Administration page, select **Factors**.

   The Factors page appears.

3. Click **Create** to display the Create Factor page.

4. Enter the following information:

   - **Name:** Enter Module.
   - **Description:** Enter Factor to find applications that can access Oracle Database.
   - **Factor Type:** From the list, select **Application**.
   - **Factor Identification:** Select **By Method**.

- **Evaluation:** Select **For Session**.
- **Factor Labeling:** Select **By Self**.
- **Retrieval Method:** Enter the following retrieval method:

```
UPPER(SYS_CONTEXT('USERENV', 'MODULE'))
```

- **Validation Method:** Leave blank.
- **Assignment Rule Set:** From the list, select **<Non Selected>**.
- **Audit Options:** Select **Never**.
- **Error Options:** Select **Show Error Message**.

5. Click **OK**.

   Oracle Database Vault creates the factor. You now can query for the factor using the same syntax you would use to query the default factors, described "Oracle Database Vault PL/SQL Factor Functions" on page 14-5.

6. In SQL*Plus, perform a quick test of the Module factor.

   For example:

```
sqlplus lbrown_dvowner
Enter password: password

SELECT DVF.F$MODULE FROM DUAL;
```

   The following output should appear:

```
F$MODULE
------------------------------
SQLPLUS.EXE
```

   Do not exit SQL*Plus. You must use it later on when you test the factor components.

Step 3: Create the Limit SQL*Plus Access Rule and Rule Set

1. In Oracle Database Vault Administrator, return to the Administration page.
2. Under Administration, select **Rule Sets**.

   The Rule Sets page appears.

3. Click **Create** to display the Create Rule Set page.
4. Enter the following settings:
   - **Name:** Enter `Limit SQL*Plus Access`.
   - **Description:** Enter `Rule set to limit access to SQL*Plus`.
   - **Status:** Select **Enabled**.
   - **Evaluation Options:** Select **All True**.
   - **Audit Options:** Select **Audit Disabled**.
   - **Error Handling Options:** Select **Show Error Message**.
   - **Fail Code, Fail Message:** Leave blank.
   - **Custom Event Handler Option:** Select **Handler Disabled**.
   - **Custom Event Handler Logic:** Leave blank.

5. Click **OK**.

   The Rule Sets page appears.

6. Select the Limit SQL*Plus rule set and then click **Edit**.

   The Edit Rule Set page appears.

7. Under Rules Associated To The Rule Set, click **Create**.

   The Create Rule page appears.

8. Enter the following settings:

   - **Name:** Enter `Prevent non-admin access to SQL*Plus`.
   - **Rule Expression:** Enter the following rule expression:

     ```
     DVF.F$MODULE != 'SQL*PLUS' AND DVF.F$SESSION_USER IN ('LBROWN_DVOWNER',
     'AMALCOLM_DVACCTMGR', 'SYS', 'SYSTEM')
     ```

     Replace `LBROWN_DVOWNER` and `AMALCOLM_DVACCTMGR` with the user account names that you use for the Database Vault Owner and Database Vault Account Manager accounts. You *must* enter the user account names in upper case letters, because that is how the database stores user account names.

     This expression tells Oracle Database Vault only to allow these four users to use SQL*Plus. Another way to write the expression is to tell Database Vault to simply exclude certain users from using SQL*Plus. That way, everyone else has access to SQL*Plus. For example, to exclude users `JSMITH` and `TSMITH` from using SQL*Plus, you would create this expression:

     ```
     DVF.F$MODULE != 'SQL*PLUS' AND DVF.F$SESSION_USER NOT IN ('JSMITH',
     'TSMITH')
     ```

     However, for this tutorial, use the first expression, in which only the four administrative users are allowed to use SQL*Plus.

9. Ensure that you have entered this rule expression *exactly* as shown in Step 8. If you enter it incorrectly, you cannot log in to SQL*Plus.

10. Click **OK**.

Step 4: Create the CONNECT Command Rule

The CONNECT command rule controls the `CONNECT` SQL statement. It also applies to logging into SQL*Plus from the command line or other tools your site may use to access SQL*Plus.

1. In Oracle Database Vault, return to the Administration page.

2. Select **Command Rules**.

   The Command Rules page appears.

3. Click **Create** to display the Create Command Rule page.

4. Enter the following settings:

   - **Command:** Select **CONNECT** from the list.
   - **Status:** Select **Enabled**.
   - **Object Owner, Object Name:** Set to **%** so that the command rule applies to anyone who logs in.
   - **Rule Set:** Select **Limit SQL*Plus Access** from the list.

5. Click **OK**.

Step 5: Test the Ad Hoc Tool Access Restriction

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

1. In SQL*Plus, try to connect as user `SCOTT`:

   ```
   CONNECT SCOTT
   Enter password: password
   ```

   The following output should appear:

   ```
   ERROR:
   ORA-47400: Command Rule violation for CONNECT on LOGON

   Warning: You are no longer connected to ORACLE.
   ```

   User `SCOTT` should be prevented from using SQL*Plus.

2. Now try to connect as user `SYSTEM`:

   ```
   CONNECT SYSTEM
   Enter password: password
   Connected.
   ```

   User `SYSTEM` should be able to log in to SQL*Plus. So should `SYS`, the Database Vault Owner account, and the Database Vault Account Manager account.

If the Test Fails

If you cannot log in to SQL*Plus as `SYSTEM` (or as any of the other administrative users listed in your rule expression), then are be prevented from using both SQL*Plus and Oracle Database Vault Administrator. But do not fear, you can remedy the problem as follows:

1. Temporarily disable Oracle Database Vault.

   See Appendix B, "Disabling and Enabling Oracle Database Vault" for instructions on disabling Oracle Database Vault. After you disable Oracle Database Vault, you can log in to SQL*Plus.

2. Log in to SQL*Plus as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   For example:

   ```
   CONNECT lbrown_dvowner
   Enter password: password
   ```

3. Enter the following statement to drop the CONNECT command rule.

   ```
   EXEC DVSYS.DBMS_MACADM.DELETE_COMMAND_RULE ('CONNECT', '%', '%');
   ```

   Even though you have disabled Oracle Database Vault, you still can use its PL/SQL Packages and Database Vault Administrator.

4. Exit SQL*Plus.

5. Reenable Oracle Database Vault.

See Appendix B, "Disabling and Enabling Oracle Database Vault" for instructions on enabling Oracle Database Vault.

6. In Oracle Database Vault Administrator, check the rule expression for any errors and then correct them. Recreate the CONNECT command rule, and then test it.

Step 6: Remove the Components for This Tutorial

1. In Database Vault Administrator, return to the Administrator page.
2. Select **Command Rules**.
3. In the Command Rules page, select the CONNECT command rule, and then click **Remove**. Select **Yes** in the Confirmation page.
4. In the Administrator page, select **Rule Sets**.
5. In the Rule Sets page, select the Limit SQL*Plus Access rule set and click **Remove**. Select **Yes** in the Confirmation page.
6. In the Administration page, select **Factors**.
7. In the Factors page, select the Module factor, and then click **Remove**. Select **Yes** in the Confirmation page.
8. As the Database Vault Owner, log in to SQL*Plus and then remove the Prevent non-admin access to SQL*Plus rule.

   For example:

```
CONNECT lbrown_dvowner
Enter password: password

EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Prevent non-admin access to SQL*Plus');
```

9. If necessary, connect as the Oracle Database Vault Account Manager and then lock and expire the `SCOTT` account.

   For example:

```
CONNECT amalcolm_dvacctmgr
Enter password: password

ALTER USER SCOTT ACCOUNT LOCK PASSWORD EXPIRE;
```

Tutorial: Restricting User Activities Based on Session Data

You can use factor identity mapping to set session-based user restrictions for database activities. For example, suppose you wanted to restrict administrative access to a database using the following criteria:

- Ensure that the administrator is accessing the database from the correct IP address.
- Limit the database access to the standard business hours of the administrator.

This type of configuration is useful for restricting different types of administrators: not only local, internal administrators, but offshore and contract administrators as well.

In this tutorial, you modify the Domain factor to include identities for a secure and non-secure network access, which are based on the IP address of the computer the administrator is using. If the administrator tries to perform an action outside the standard working hours or from a different IP address, then Oracle Database Vault prevents him from doing so.

In this tutorial:

- Step 1: Create an Administrative User
- Step 2: Add Identities to the Domain Factor
- Step 3: Map the Domain Factor Identities to the Client_IP Factor
- Step 4: Create a Rule Set to Set the Hours and Select the Factor Identity
- Step 5: Create a Command Rule That Uses the Rule Set
- Step 6: Test the Factor Identity Settings
- Step 7: Remove the Components for This Tutorial

Step 1: Create an Administrative User

1. In SQL*Plus, log in as a user who has been granted the `DV_ACCTMGR` role, and then create the user account `mwaldron`.

   For example:

   ```
   sqlplus amalcolm_dvacctmgr
   Enter password: password

   CREATE USER mwaldron IDENTIFIED BY password;
   ```

   Replace *`password`* with a password that is secure.

2. Connect as `SYS` using the `SYSDBA` privilege, and then grant user `mwaldron` DBA privileges.

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

   GRANT CREATE SESSION, DBA TO mwaldron;
   ```

Step 2: Add Identities to the Domain Factor

1. Log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

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

2. In the Administration page, select **Factors**.

   The Factors page appears.

3. Select the Domain factor and then select **Edit**.

   The Domain factor is the parent factor.

4. Under Identities, select **Create**.

5. In the Create Identity page, enter the following information:

   - **Value:** Enter `HIGHLY SECURE INTERNAL NETWORK`
   - **Trust Level:** Select **Very Trusted**

6. Click **OK**.

7. In the Edit Factor:Domain page, create a second identity called NOT SECURE, and then set its trust level to Untrusted.

Step 3: Map the Domain Factor Identities to the Client_IP Factor

1. In Oracle Database Vault Administrator, in the Edit Factor: Domain page, select the HIGHLY SECURE INTERNAL NETWORK identity and then select **Edit**.
2. Under Map Identity, select **Create**.
3. In the Create Identity Map page, enter the following information:
   - **Contributing Factor**: Select **Client_IP** to be the child factor.
   - **Map Condition**: Select **Equal**, and then in the **Low Value** box, enter the IP address for the Virtual Machine, for example, `192.0.2.12`. (This is the computer that user `maldron` uses. For this tutorial, you can enter the IP address of your own computer. If you are using Microsoft Windows, use the IP address assigned to the Loopback Adapter.)
4. Click **OK**, and then click **OK** again to return to the Edit Factor: Domain page.
5. Create the following two identity maps for the NOT SECURE identity:

Child Factor Name	Operation Value	Operand 1	Operand 2
`Client_IP`	`Less`	`192.0.2.5`	(Leave blank)
`Client_IP`	`Greater`	`192.0.2.20`	(Leave blank)

   The identity maps in the NOT SECURE identity are in a range of IP addresses outside the IP address that user `mwaldron` uses (192.0.2.12). The IP addresses here must be in any range *outside* `mwaldron`'s IP address.

   This identity mapping creates the following condition: If the user logs in from the correct IP address, then Oracle Database Vault decides that the connection is secure, through the HIGHLY SECURE INTERNAL NETWORK identity. However, if the user logs in from an IP address that is less than 192.0.2.5 or greater than 192.0.2.20, then the connection is deemed not secure, through the NO SECURE identity.
6. Click **OK** to return to the Edit Factor: Domain page, then click **OK** again to return to the Factors page.
7. Test the factor identities.

   First, in SQL*Plus, connect as user `mwaldron` but do not specify a database instance.

```
CONNECT mwaldron
Enter password: password

SELECT DVF.F$CLIENT_IP FROM DUAL;
```

   The following output should appear:

```
F$CLIENT_IP
-----------------------------------
```

   Next:

```
SELECT DVF.F$DOMAIN FROM DUAL;
```

   The following output should appear:

```
F$DOMAIN
-----------------------------------
NOT SECURE
```

Because user `mwaldron` is not connecting directly to the database instance, Oracle Database Vault does not recognize the IP address from which he is connecting. In this case, Oracle Database uses the IPC protocol to perform the connection, which sets the IP value to null. Therefore, the identity for this connection is set to NOT SECURE.

Now connect to SQL*Plus by specifying the database instance (for example, `orcl`) and then check the factor identities again:

```
CONNECT mwaldron@orcl
Enter password: password

SELECT DVF.F$CLIENT_IP FROM DUAL;
```

The following output should appear:

```
F$CLIENT_IP
-----------------------------------
192.0.2.12
```

Next:

```
SELECT DVF.F$DOMAIN FROM DUAL;
```

The following output should appear:

```
F$DOMAIN
-----------------------------------
HIGHLY SECURE INTERNAL NETWORK
```

Now that user `mwaldron` is connecting to the `orcl` database instance, his IP address is recognized. This is because the database uses the TCP protocol, so now the host IP value can be populated appropriately. Because the IP address is within the correct range, the factor identity is set to HIGHLY SECURE INTERNAL NETWORK.

Step 4: Create a Rule Set to Set the Hours and Select the Factor Identity

1. In Oracle Database Vault, return to the Administration page.
2. Select **Rule Sets**.
3. In the Rule Sets page, select **Create**.
4. In the Create Rule Set page, enter the following settings:
   - **Name:** Enter `Internal DBA Standard Working Hours`.
   - **Status:** Select **Enabled**.
   - **Evaluation Options:** Select **All True**.

   Leave the remaining settings at their defaults.
5. Click **OK**.
6. In the Rule Sets page, select the Internal DBA Standard Working Hours rule set, and then select **Edit**.
7. In the Edit Rule Set: Internal DBA Standard Working Hours page, under Rules Associated To The Rule Set, select **Create**.
8. In the Create Rule page, create the following rules:

- **Name:** `Internal DBA`

  Rule Expression: `DVF.F$SESSION_USER='MWALDRON'`

  (When you create an expression with a user name, enter the user name in upper case letters, because that is how the database stores user names.)

- **Name:** `Internal Network Only`

  Rule Expression: `DVF.F$DOMAIN='HIGHLY SECURE INTERNAL NETWORK'`

- **Name:** `Week Day`

  Rule Expression: `TO_CHAR(SYSDATE, 'D') BETWEEN '2' AND '6'`

- **Name:** `Week Working Day Hours`

  Rule Expression: `TO_CHAR(SYSDATE, 'HH24') BETWEEN '08' AND '19'`

9. Click **OK** to return to the Rule Sets page.

Step 5: Create a Command Rule That Uses the Rule Set

1. In Oracle Database Vault Administrator, return to the Administration page.
2. Select **Command Rules**, and in the Command Rules page, select **Create**.
3. In the Create Command Rule page, enter the following settings:
   - **Command:** Select **CREATE TABLE** from the list.
   - **Rule Set:** Select **Internal DBA Standard Working Hours** from the list.

   Leave the remaining settings at their defaults.
4. Click **OK**.

Step 6: Test the Factor Identity Settings

Test the settings by resetting the system clock, logging in as the `mwaldron` administrative user, and then trying to create a table.

1. Set the system time to 9 p.m.

   UNIX: Log in as root and use the date command to set the time. For example, assuming the date today is December 13, 2007, you would enter the following:

   ```
   su root
   Password: password

   date 12132109
   ```

   Windows: Double-click the clock icon, which is typically at the lower right corner of the screen. In the Date and Time Properties window, set the time to 9 p.m., and then click **OK**.

2. In SQL*Plus, connect as user `mwaldron` and try to create a table. In the following, replace `orcl` with the name of your database instance.

   ```
   CONNECT mwaldron@orcl
   Enter password: password

   CREATE TABLE TEST (num number);
   ```

The following output should appear:

```
ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-47400: Command Rule violation for create table on MWALDRON.TEST
ORA-06512: at "DVSYS.AUTHORIZE_EVENT", line 55
ORA-06512: at line 31
```

Because user `mwaldron` is create a table outside working hours, Database Vault prevents him.

3. Reset the system time back to the local time.
4. In SQL*Plus, as user `mwaldron`, try to create the table again.

```
CREATE TABLE TEST (num number);

Table created.

DROP TABLE TEST;
Table dropped.
```

Now that user `maldron` is working during his local hours and from the IP address associated with the HIGHLY SECURE INTERNAL NETWORK identity, he can create tables.

5. Reconnect as user `mwaldron` but without adding the database instance name to the connection command, and then try to select from the `OE.ORDERS` table again.

```
CONNECT mwaldron
Enter password: password

CREATE TABLE TEST (num number);
```

The following output should appear:

```
ERROR at line 1:
ORA-00604: error occurred at recursive SQL level 1
ORA-47400: Command Rule violation for create table on MWALDRON.TEST
ORA-06512: at "DVSYS.AUTHORIZE_EVENT", line 55
ORA-06512: at line 31
```

Even though user `mwaldron` is trying to create a table during the correct time, he cannot because is not directly logged in to the `orcl` database instance. Oracle Database Vault deems him to be using the NOT SECURE identity, and then denies him access.

Step 7: Remove the Components for This Tutorial

1. Log in to SQL*Plus as the `DV_ACCTMGR` user and drop user `mwaldron`.

```
sqlplus amalcolm_dvacctmgr
Enter password: password

DROP USER mwaldron CASCADE;
```

2. Remove the CREATE TABLE command rule.

   Return the Administration page and select **Command Rules**. Select the CREATE TABLE command rule and then select **Remove**. In the Confirmation page, select **Yes**.

3. Remove the Internal DBA Standard Working Hours rule set.

In Oracle Database Vault Administrator, select **Rule Sets** in the Administration page. In the Rule Sets page, select the Internal DBA Standard Working Hours rule set, and then select **Remove**. In the Confirmation page, select **Yes**.

4. In SQL*Plus, delete the rules associated with the Internal DBA Standard Working Hours rule set.

```
CONNECT lbrown_dvowner
Enter password: password

EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Internal DBA');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Internal Network Only');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Week Day');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Week Day Working Hours');
COMMIT;
```

5. Remove the HIGHLY SECURE INTERNAL NETWORK and NOT SECURE factor identities from the Domain factor.

   Return to the Administration page and select **Factors**. Select the Domain factor, select **Edit**, and under Identities, remove the HIGHLY SECURE INTERNAL NETWORK and NOT SECURE factor identities. In the Confirmation page, select **Yes**.

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 9-6 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 unnecessary factors by deleting them. (Oracle Database Vault does not use any of these factors internally, so you can remove them if you do not need them.)

If you have a large number of users or if your application server frequently must create and destroy connections, the resources used can affect system performance. You can delete the unnecessary factors.

You can check system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager 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 and Data Dictionary Views

Table 7–1 lists Oracle Database Vault reports that are useful for analyzing factors and their identities. See Chapter 16, "Oracle Database Vault Reports" for information about how to run these reports.

Table 7–1 Reports Related to Factors and Their Identities

Report	Description
"Factor Audit Report" on page 16-4	Audits factors (for example, to find factors that failed to be evaluated)
"Factor Configuration Issues Report" on page 16-3	Lists configuration issues, such as disabled or incomplete rule sets, or to audit issues that may affect the factor
"Factor Without Identities Report" on page 16-3	Lists factors that have had no identities assigned yet

Table 7–1 (Cont.) Reports Related to Factors and Their Identities

Report	Description
"Identity Configuration Issues Report" on page 16-3	Lists factors that have invalid label identities or no map for the identity
"Rule Set Configuration Issues Report" on page 16-4	Lists rule sets that have no rules defined or enabled, which may affect the factors that use them

Table 7–2 lists data dictionary views that provide information about existing factors and factor identities.

Table 7–2 Data Dictionary Views Used for Factors and Factor Identities

Data Dictionary View	Description
"DBA_DV_FACTOR View" on page 10-15	Lists the existing factors in the current database instance
"DBA_DV_FACTOR_LINK View" on page 10-16	Shows the relationships of each factor whose identity is determined by the association of child factors
"DBA_DV_FACTOR_TYPE View" on page 10-17	Lists the names and descriptions of factor types used in the system
"DBA_DV_IDENTITY View" on page 10-17	Lists the identities for each factor
"DBA_DV_IDENTITY_MAP View" on page 10-18	Lists the mappings for each factor identity

8

Configuring Secure Application Roles for Oracle Database Vault

This chapter contains:

- What Are Secure Application Roles in Oracle Database Vault?
- Creating and Editing Secure Application Roles
- Securing a Secure Application Role
- Deleting a Secure Application Role
- How Secure Application Roles Work
- Tutorial: Granting Access with Database Vault Secure Application Roles
- How Secure Application Roles Affect Performance
- Related Reports and Data Dictionary View

What Are Secure Application Roles in Oracle Database Vault?

In Oracle Database Vault, you can create a secure application role that you enable with an Oracle Database Vault rule set. Regular Oracle Database secure application roles are enabled by custom PL/SQL procedures. You use secure application roles to prevent users from accessing data from outside an application. This forces users to work within the framework of the application privileges that have been granted to the role.

The advantage of basing database access for a role on a rule set is that you can store database security policies in one central place, as opposed to storing them in all your applications. Basing the role on a rule set provides a consistent and flexible method to enforce the security policies that the role provides. In this way, if you must update the security policy for the application role, you do it in one place, the rule set. Furthermore, no matter how the user connects to the database, the result is the same, because the rule set is bound to the role. All you need to do is to create the role and then associate it with a rule set. The associated rule set will validate the user who is trying to enable the role.

You can run reports on secure application roles that you create in Oracle Database Vault. See "Related Reports and Data Dictionary View" on page 8-9 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 chapters:

- Chapter 11, "Using the DVSYS.DBMS_MACADM Package"
- Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces"

Creating and Editing Secure Application Roles

Follow these steps:

1. Create a rule set that contains at least one rule to set the conditions for allowing or disallowing the user to enable the role.

   When you create the underlying rule for the rule set, remember that the rule should validate the user who is trying to enable the role. The rule should validate the user who is trying to log in. See Chapter 5, "Configuring Rule Sets" for more information about rule sets.

2. Log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

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

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

4. 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:

     ```
     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 Procedure" on page 12-2 for more information about the `SET_ROLE` function.

5. 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 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:** Enables the role to be available for use. That is, users are allowed to call the `DVSYS.DBMS_MACSEC_ROLES.SET_ROLE` function to try to enable the role. Note that whether or not the role will be enabled depends on the evaluation result of the associated rule set.

    See "SET_ROLE Procedure" on page 12-2 for more information about this function.
  - **Disabled:** Disables the role from being available for use. The `DVSYS.DBMS_MACSEC_ROLES.SET_ROLE` function will not be able to enable the role.

  See "Oracle Database Vault PL/SQL Packages" on page 14-18 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 may try to use the `DROP ROLE` SQL statement 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 Data Dictionary Views" on page 10-11 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 Procedures Within DVSYS.DBMS_MACADM" on page 11-45 for more information.

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

   See "SET_ROLE Procedure" on page 12-2 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.

Tutorial: Granting Access with Database Vault Secure Application Roles

In this tutorial, you restrict the `SELECT` SQL statement on the `ORDERS` table in the `OE` schema to a specific set of users. Furthermore, these users can only perform these statements on the `OE.ORDERS` table from within the office, not from a remote connection. To accomplish this, you create an Oracle Database Vault secure application role that is enabled for the user only if the user passes the checks enforced by the rule set that you associate with the secure application role.

In this tutorial:

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

Step 1: Create Users for This Tutorial

1. Log in to SQL*Plus as a user who has been granted the `DV_ACCTMGR` role.

   For example:

   ```
   sqlplus amalcolm_dvacctmgr
   Enter password: password
   ```

2. Create the following user accounts:

   ```
   CREATE USER eabel IDENTIFIED BY password;
   CREATE USER ahutton IDENTIFIED BY password;
   CREATE USER ldoran IDENTIFIED BY password;
   ```

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

3. Connect as SYS using the SYSDBA privilege, and then grant these users the CREATE SESSION privilege.

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

GRANT CREATE SESSION TO eabel, ahutton, ldoran;
```

Step 2: Enable the OE User Account

1. In SQL*Plus, connect as the Database Vault Account Manager.

   For example:

```
CONNECT amalcolm_dvacctmgr
Enter password: password
```

2. Check the account status of the OE account.

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

3. If the OE account is locked and expired, unlock it and assign it a new password.

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

Step 3: Create the Rule Set and Its Rules

1. Log in to Oracle Database Vault Administrator as a user who has been granted the DV_OWNER or DV_ADMIN role.

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

2. In the Administration page, select **Rule Sets**.

   The Rule Sets page appears.

3. Click **Create**.

   The Create Rule Set page appears.

4. Enter the following information:

   - **Name:** Can Modify Orders
   - **Description:** Rule set to control who can modify orders in the OE.ORDERS table

5. Leave the remaining settings and their defaults, and then click **OK**.

6. In the Rule Sets page, select the Can Modify Orders rule set, and then click **Edit**.

   The Edit Rule Set: Can Modify Orders page appears.

7. Scroll to the bottom of the page and under Rules Associated To The Rule Set, click **Create**.

8. Create the following two rules:

Rule Name	Rule Expression
Check IP Address	DVF.F$CLIENT_IP = '192.0.2.12'

```
Check Session User DVF.F$SESSION_USER IN ('EABEL', 'AHUTTON')
```

For the Check IP Address rule, replace 123.45.67.89 with the IP address for your own computer. In a real-world scenario, you would create an expression that includes all the IP addresses for the users who should be allowed access.

Both of these rules use the default factors Client_IP and Session_User. See "Default Factors" on page 7-2 for more information about these factors. If these factors have been removed or modified, you can use the following rule expressions instead:

- **Check IP Address:** `UPPER(SYS_CONTEXT('USERENV','IP_ADDRESS')) = '192.0.2.12'`
- **Check Session User:** `UPPER(SYS_CONTEXT('USERENV','SESSION_USER')) IN ('EABEL', 'AHUTTON')`

9. Ensure that the **Status** setting for the Can Modify Orders table is set to **Enabled** and **Evaluation Options** is set to **All True**.
10. Click **OK**.

Step 4: Create the Database Vault Secure Application Role

1. In Oracle Database Vault, return to the Administration page.
2. Under Administration, select **Secure Application Roles**.

   The Secure Application Roles page appears.
3. Click **Create**.

   The Create Role page appears.
4. In the **Role** box, enter `ORDERS_MGMT` to name the role.
5. Under Rule Set, select **Can Modify Orders**.
6. Click **OK**.

At this stage, the Database Vault secure application role and its associated rule set are created, though the role does not yet have any privileges.

Step 5: Grant the SELECT Privilege to the Secure Application Role

1. In SQL*Plus, connect as user `OE`.

   ```
   CONNECT OE
   Enter password: password
   ```

2. Grant the `SELECT` privilege to the `ORDERS_MGMT` Database Vault Secure application role.

   ```
   GRANT SELECT ON ORDERS TO ORDERS_MGMT;
   ```

Step 6: Test the Database Vault Secure Application Role

1. In SQL*Plus, connect directly to the database as user `eabel`.

   ```
   CONNECT eabel@orcl
   Enter password: password
   ```

   Replace `orcl` with the name of your database instance.
2. Set the `ORDERS_MGMT` role.

```
EXEC DVSYS.DBMS_MACSEC_ROLES.SET_ROLE('ORDERS_MGMT');
```

Typically, you would embed this call in the application that the user logs in to.

3. Select from the `OE.ORDERS` table.

```
SELECT COUNT(*) FROM OE.ORDERS;
```

The following output should appear:

```
  COUNT(*)
----------
       105
```

Because user `eabel` is logging directly into the database from the correct IP address and is listed as a valid session user, she can select from the `OE.ORDERS` table. If user `ahutton` logs in to SQL*Plus in the same manner, she also can select from the `OE.ORDERS` table.

4. Reconnect as user `eabel` without specifying the database instance, and then try to select from the `OE.ORDERS` table again.

```
CONNECT eabel
Enter password: password

EXEC DVSYS.DBMS_MACSEC_ROLES.SET_ROLE('ORDERS_MGMT');
```

The following output should appear:

```
Error at line 1:
ORA-47305: Rule Set Violation on SET ROLE (Can Modfiy Orders)
...
```

Next:

```
SELECT COUNT(*) FROM OE.ORDERS;
```

The following output should appear:

```
ERROR at line 1:
ORA-00942: table or view does not exist
```

Even though user `eabel` is a valid user, she has violated the Check IP Address rule in the rule set, so she is not granted the `ORDERS_MGMT` role. The only way for the IP address to be recognized is to connect by specifying the database instance, as user `eabel` did in Step 1. (For an explanation about how this works, see Step 7 in "Step 3: Map the Domain Factor Identities to the Client_IP Factor" on page 7-24, in Chapter 7.)

5. Connect as user `ldoran` and then enter the following statements:

```
EXEC DVSYS.DBMS_MACSEC_ROLES.SET_ROLE('ORDERS_MGMT');
SELECT COUNT(*) FROM OE.ORDERS;
```

Because user `ldoran` is not a valid user, she is not granted the `ORDERS_MGMT` role. Therefore, she cannot select from the `OE.ORDERS` table.

Step 7: Remove the Components for This Tutorial

1. Log into SQL*Plus as the Database Vault Owner.

   For example:

   ```
   CONNECT lbrown_dvowner
   Enter password: password
   ```

2. Delete the ORDERS_MGMT secure application role.

   ```
   EXEC DVSYS.DBMS_MACADM.DELETE_ROLE('ORDERS_MGMT');
   ```

3. Enter the following commands in the order shown to remove the Can Modify Orders rule set.

   ```
   EXEC DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('Can Modify Orders', 'Check IP
   Address');
   EXEC DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('Can Modify Orders', 'Check
   Session User');
   EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Check IP Address');
   EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Check Session User');
   EXEC DVSYS.DBMS_MACADM.DELETE_RULE_SET('Can Modify Orders');
   COMMIT;
   ```

4. Connect as the Database Vault Account Manager and drop the users.

   For example:

   ```
   CONNECT amalcolm_dvacctmgr
   Enter password: password

   DROP USER eabel;
   DROP USER ahutton;
   DROP USER ldoran;
   ```

5. If uneccessary, lock and expire the OE user account.

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

How Secure Application Roles Affect Performance

You can check system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager 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 and Data Dictionary View

Table 8–1 lists Oracle Database Vault reports that are useful for analyzing Oracle Database Vault secure application roles. See Chapter 16, "Oracle Database Vault Reports" for information about how to run these reports.

Table 8–1 Reports Related to Secure Application Roles

Report	Description
"Secure Application Role Audit Report" on page 16-5	Lists audit records generated by the Oracle Database Vault secure application role-enabling operation. To generate this type of audit record, enable auditing for the rule set associated with the role.
"Secure Application Configuration Issues Report" on page 16-4	Lists secure application roles that have nonexistent database roles, or incomplete or disabled rule sets
"Rule Set Configuration Issues Report" on page 16-4	Lists rule sets that have no rules defined or enabled, which may affect the secure application roles that use them
"Powerful Database Accounts and Roles Reports" on page 16-9	Provides information about powerful database accounts and roles

You can use the `DBA_DV_ROLE` data dictionary view to find the Oracle Database Vault secure application roles used in privilege management. See "DBA_DV_ROLE View" on page 10-23 for more information.

9

Integrating Oracle Database Vault with Other Oracle Products

This chapter contains:

- Using Oracle Database Vault with Oracle Enterprise Manager Grid Control
- 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
- Using Oracle Database Vault with Oracle Recovery Manager

Using Oracle Database Vault with Oracle Enterprise Manager Grid Control

This section contains:

- Setting the Database Vault Administrator URL in Oracle Enterprise Manager
- Propagating Oracle Database Vault Policies to Other Databases
- Using Enterprise Manager Grid Control Alerts for Oracle Database Vault Policies
- Using Oracle Database Vault-Specific Reports in Enterprise Manager Grid Control

Setting the Database Vault Administrator URL in Oracle Enterprise Manager

You can configure Database Control or Grid Control to use a specific Database Vault Administrator URL.

1. Access Oracle Database Vault Home page from Oracle Enterprise Manager.

   See "Accessing Oracle Database Vault Pages from Oracle Enterprise Manager" on page 3-1.

   If you want to save the Database Vault URL:

   - **For Database Control:** Ensure that in addition to being granted the `DV_OWNER` or `DV_ADMIN` role, that you are also an Enterprise Manager administrator.
   - **For Grid Control:** Ensure that you have been granted the `OPERATOR` privilege on the target database.

2. Access the Database Vault Administrator URL page.

   From the Database Vault Home page, click the **Administration** tab, and then under Policy Administration, click the **Launch Database Vault Administrator** link.

   The Database Vault Administrator URL page appears.

3. In the Database Vault Administrator URL field, enter the Database Vault Administrator URL that you want to use.

   For example:

   ```
   https://myserver.us.example.com:1148/dva
   ```

4. To test the URL, click **Launch to test URL** button.

5. Click the **Save** button.

Propagating Oracle Database Vault Policies to Other Databases

If you have Oracle Database Vault installed in an Oracle Enterprise Manager Grid Control Release 10.2.0.5 environment, then you can propagate Database Vault policies to other Database Vault-protected databases. You cannot use Grid Control to create Database Vault policies or perform the actions normally provided for in Database Vault Administrator. If you want to perform these functions, then use Database Vault Administrator.

To propagate Database Vault policies to other databases:

1. Access Oracle Database Vault Administrator from Oracle Enterprise Manager, and log in to Database Vault as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Ensure that you select the database that contains the policies that you want to propagate.

   See "Accessing Oracle Database Vault Pages from Oracle Enterprise Manager" on page 3-1.

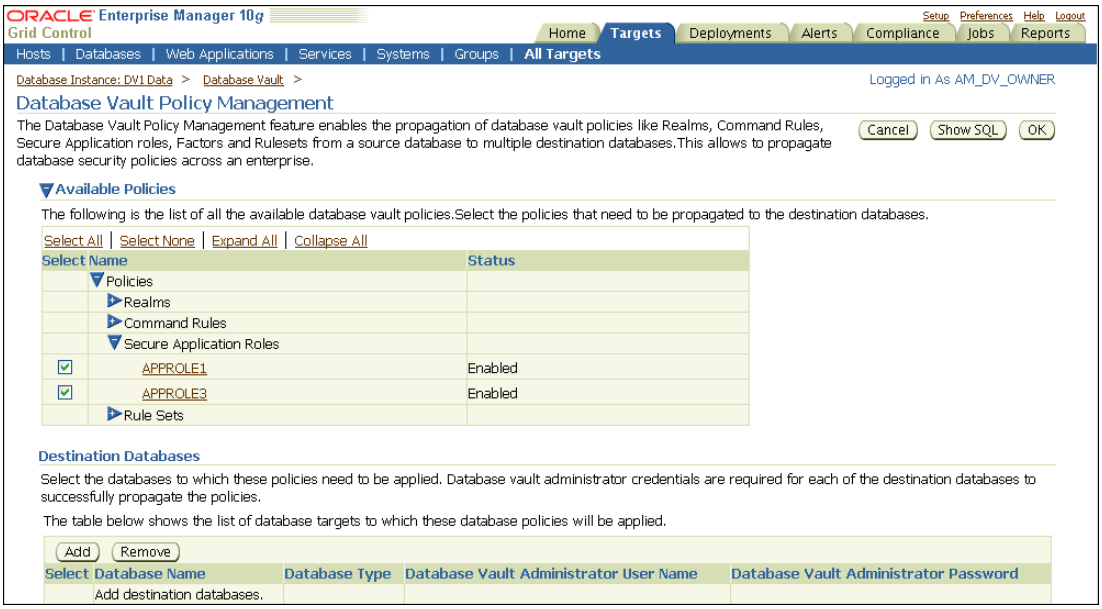
2. From the Database Vault home page, select the **Administration** subpage.

3. In the Administration page, under Policy Propagation, select the **Database Vault Policy Propagation** link.

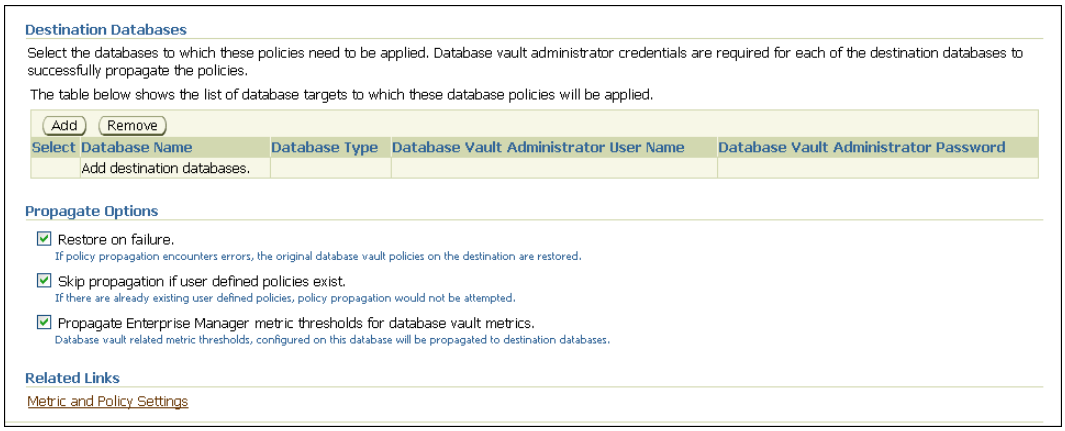
   The Available Policies area in the Policy Propagation subpage lists a summary of the Oracle Database Vault policies that were created for the database that you selected in Step 1. From here, you can propagate these policies to another database.

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

   By default, all policies are selected.



5. Under Destination Databases, click the **Add** button.



6. Under Search and Select: Database Vault Enabled Destination Databases, search for the destination databases, and then select each database to which you want to propagate the policies. Then click the **Select** button.
7. Under Destination Databases, do the following:
   a. Under Apply credentials across destination database(s), enter the user name and password of the administrator of the Database Vault database that contains the policies you want to propagate.

      This feature applies the Database Vault administrator's user name and password to all of the selected destination databases.

   b. Select each database to which you want to propagate the policies.
   c. Enter the Database Vault administrator user name and password for each database.
   d. Click the **Apply** button.
8. In the Propagate Options page, select from the following options.

Any changes made to the seeded realms, command rules, rule sets, and so on will not be propagated to the destination databases. Only custom-created data are propagated.

- **Restore on failure:** If the policy propagation encounters errors, then the propagation is rolled back. That is, the original policies on the destination database are restored. If you do not select this option, then the policy propagation on the destination database continues and ignores any errors.
- **Skip propagation if user defined policies exist:** If the destination databases already have the user-defined policies, then the policy propagation is not attempted. If you do not select this option, then regardless of whether user-defined policies exist on the destination database, all the existing policies are cleared, and the policies from the source database are applied to the destination database.
- **Propagate Enterprise Manager metric thresholds for database vault metrics:** If the source database has Oracle Database Vault metric thresholds set, then these thresholds are also propagated to the destination databases. If you do not select this option, then only policies are propagated and not the Oracle Database Vault thresholds.

9. Click the **OK** button.
10. In the Confirmation window, click **OK**.

    A message indicating success or failure appears. If the propagation succeeds, then the policies are active right away in their destination databases.

Using Enterprise Manager Grid Control Alerts for Oracle Database Vault Policies

Grid Control generates Oracle Database Vault-specific alerts. To view these alerts, you must be granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role. The alerts are as follows:

- **Database Vault Attempted Realm Violations.** This alert helps the Oracle Database Vault security analyst (`DV_SECANALYST` role) to monitor violation attempts on the Database Vault database. This user can select the realms to be affected by the alert and filter these realms based on the different types of attempts by using error codes. You can enable this metric from the Metrics and Policy Settings page. By default, the attempted realm violations are collected every 24 hours.
- **Database Vault Attempted Command Rule Violations.** The functionality for this alert is the same as for Database Vault Attempted Realm Violations, except that it focuses on violations on command rules.
- **Database Vault Realm Configuration Issues.** This metric tracks and raises an alert if users misconfigure realms. This metric is enabled when you install Oracle Database vault, and by default it collects data every one hour.
- **Database Vault Command Rule Configuration Issues.** This functionality for this alert is that same as Database Vault Realm Configuration Issues, except that it focuses on configuration changes to command rules.
- **Database Vault Policy Changes.** This metric raises an alert on any change to any Database Vault policy, such as policies for realms and command rules. It provides a detailed policy changes report.

Using Oracle Database Vault-Specific Reports in Enterprise Manager Grid Control

From the Database Vault home page, you can find information about the following types of violations:

- Top five attempted violations on realm and command rule
- Top five attempted violations by database users and client host
- Time series-based graphical reports on attempted violations for more detailed analysis

To have full access to the Database Vault reports, you must log in to Database Vault Administrator as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

See Also: Chapter 16, "Oracle Database Vault Reports"

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.example.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

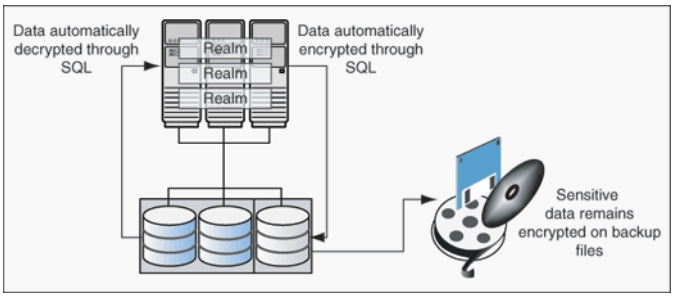
Oracle Database Vault works with Transparent Data Encryption (TDE). With Transparent Data Encryption, an application administrator can use a single one line command to alter a table and encrypt a column. Subsequent inserts into that table column are written to disk encrypted transparent to the SQL. This 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 Transparent Data Encryption enabled, Oracle Database Vault works with it seemlessly and without any additional configuration. Transparent Data Encryption 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



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

How Oracle Database Vault Is Integrated with Oracle Label Security

When you integrate Oracle Database Vault with Oracle Label Security, it means that you can assign an Oracle Label Security label to an Oracle Database Vault factor identity.

In Oracle Label Security, you can restrict access to records in database tables or PL/SQL programs. For example, Mary may be able to see data protected by the

HIGHLY SENSITIVE label, an Oracle Label Security label on the `EMPLOYEE` table that includes records that should have access limited to certain managers. Another label can be PUBLIC, which allows more open access to this data.

In Oracle Database Vault, you can create a factor called Network, for the network on which the database session originates, with the following identities:

- **Intranet:** Used for when an employee is working on site within the intranet for your company.
- **Remote:** Used for when the employee is working at home from a VPN connection.

You then assign a maximum session label to both. For example:

- Assign the Intranet identity to the HIGHLY SENSITIVE Oracle Label Security label.
- Assign the Remote identity to the PUBLIC label.

This means that when Mary is working at home using her VPN connection, she has access only to the limited table data protected under the PUBLIC identity. But when she is in the office, she has access to the HIGHLY SENSITIVE data, because she is using the Intranet identity. "Tutorial: Integrating Oracle Database Vault with Oracle Label Security" on page 9-9 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 16-5 for more information. Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` system file, described in Appendix A, "Auditing Oracle Database Vault."

You can use the Oracle Database Vault APIs to integrate Oracle Database Vault with Oracle Label Security. See Chapter 11, "Using the 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 and Data Dictionary Views" on page 9-13 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. Ensure that 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 Oracle Database Vault Factors with Oracle Label Security Policies

Oracle Database Vault controls the maximum security clearance for a database session by merging the maximum allowable data for each label in a database session by merging the labels of Oracle Database Vault factors that are associated to an Oracle

Label Security policy. In brief, a label acts as an identifier for the access privileges of a database table row. A policy is a name associated with the labels, rules, and authorizations that govern access to table rows. See *Oracle Label Security Administrator's Guide* for more information about row labels and policies.

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 as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   "Starting Oracle Database Vault" 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 must determine the resulting label that is required when combining two data sets that have different labels. It is also necessary for situations in which you must perform queries using joins on rows with different data labels.

  For more information on these label-merging algorithms, see *Oracle Label Security Administrator's Guide*. If you want to use the `DVSYS.DBMS_MACADM` package to specify a merge algorithm, see Table 11–57, " Oracle Label Security Merge Algorithm Codes" on page 11-49 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.

Tutorial: Integrating Oracle Database Vault with Oracle Label Security

You can use Oracle Database Vault factors with Oracle Label Security and Oracle Virtual Private Database (VPD) technology to restrict access to sensitive data. You can restrict this data so that it is only exposed to a database session when the correct combination of factors exists, defined by the security administrator, for any given database session.

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

In this tutorial:

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

Step 1: Create Users for This Tutorial

1. Log in to SQL*Plus as a user who has been granted the `DV_ACCTMGR` role.

   For example:

   ```
   sqlplus amalcolm_dvacctmgr
   Enter password: password
   ```

2. Create the following users:

```
CREATE USER mdale IDENTIFIED BY password;
CREATE USER jsmith IDENTIFIED BY password;
```

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

3. Connect as user SYS with the SYSDBA privilege and then grant administrative privileges to users mdale and jsmith.

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

GRANT CREATE SESSION, DBA TO mdale, jsmith;
```

At this stage, users mdale and jsmith have identical administrative privileges.

Step 2: Create the Oracle Label Security Policy

1. In SQL*Plus, connect as the Oracle Label Security administrator, LBACSYS.

```
CONNECT LBACSYS
Enter password: password
```

If user LBACSYS is locked and expired, connect as the Database Vault Account Manager, unlock and unexpire the LBACSYS account, and then log back in as LBACSYS.

For example:

```
CONNECT amalcolm_dvacctmgr
Enter password: password

ALTER USER LBACSYS ACCOUNT UNLOCK IDENTIFIED BY password;

CONNECT LBACSYS
Enter password: password
```

2. Create a new Oracle Label Security policy:

```
EXEC SA_SYSDBA.CREATE_POLICY('PRIVACY','PRIVACY_COLUMN','NO_CONTROL');
```

3. Create the following levels for the PRIVACY policy:

```
EXEC SA_COMPONENTS.CREATE_LEVEL('PRIVACY',2000,'S','SENSITIVE');
EXEC SA_COMPONENTS.CREATE_LEVEL('PRIVACY',1000,'C','CONFIDENTIAL');
```

4. Create the PII compartment.

```
EXEC SA_COMPONENTS.CREATE_COMPARTMENT('PRIVACY',100,'PII','PERS_INFO');
```

5. Grant users mdale and jsmith the following labels:

```
EXEC SA_USER_ADMIN.SET_USER_LABELS('PRIVACY','mdale','S:PII');
EXEC SA_USER_ADMIN.SET_USER_LABELS('PRIVACY','jsmith','C');
```

User mdale is granted the more sensitive label, Sensitive, which includes the PII compartment. User jsmith gets the Confidential label, which is less sensitive.

Step 3: Create Oracle Database Vault Rules to Control the OLS Authorization

1. Connect to SQL*Plus as the Database Vault Owner.

   For example:

   ```
   CONNECT lbrown_dvowner
   Enter password: password
   ```

2. Create the following rule set:

   ```
   EXEC DVSYS.DBMS_MACADM.CREATE_RULE_SET('PII Rule Set',
    'Protect PII data from privileged users','Y',1,0,2,NULL,NULL,0,NULL);
   ```

3. Create a rule for the PII Rule Set.

   ```
   EXEC DVSYS.DBMS_MACADM.CREATE_RULE('Check OLS Factor',
    'dominates(sa_utl.numeric_label(''PRIVACY''),
     char_to_label(''PRIVACY'',''S:PII'')) = ''1''');
   ```

   Ensure that you use single quotes, as shown in this example, and not double quotes.

4. Add the Check OLS Factor rule to the PII Rule Set.

   ```
   EXEC DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET('PII Rule Set',
    'Check OLS Factor');
   ```

5. Synchronize the Check OLS factor rule.

   ```
   EXEC DVSYS.DBMS_MACADM.SYNC_RULES;
   COMMIT;
   ```

Step 4: Update the ALTER SYSTEM Command Rule to Use the Rule Set

1. As the Database Vault Owner, check the current value of the ALTER SYSTEM command rule, which is one of the default command rules when you install Oracle Database Vault.

   ```
   SELECT * FROM DVSYS.DBA_DV_COMMAND_RULE WHERE COMMAND = 'ALTER SYSTEM';
   ```

2. Make a note of these settings so that you can revert them to their original values later on.

   In a default installation, the ALTER SYSTEM command rule uses the Allow System Parameters rule set, has no object owner or name, and is enabled.

3. Update the ALTER SYSTEM command rule to include the PII Rule Set.

   ```
   EXEC DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE('ALTER SYSTEM', 'PII Rule Set', '%',
   '%', 'Y');
   ```

   This command adds the PII Rule Set to the ALTER SYSTEM command rule, applies it to all object owners and object names, and enables the command rule.

Step 5: Test the Authorizations

1. In SQL*Plus, log on as user `mdale`.

   ```
   CONNECT mdale
   Enter password: password
   ```

2. Check the current setting for the `AUDIT_TRAIL` initialization parameter.

   ```
   SHOW PARAMETER AUDIT_TRAIL
   ```

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------
audit_trail                          string      DB
```

Make a note of this setting, so that you can revert it to its original setting later on.

3. As user `mdale`, use the `ALTER SYSTEM` statement to modify the `AUDIT_TRAIL` parameter.

```
ALTER SYSTEM SET AUDIT_TRAIL=OS, EXTENDED SCOPE=SPFILE;
System altered.
```

Because user `mdale` was assigned the Sensitive label with the PII compartment, he can use the `ALTER SYSTEM` statement to modify the `AUDIT_TRAIL` system parameter.

4. Set the `AUDIT_TRAIL` parameter back to its original value, for example:

```
ALTER SYSTEM SET AUDIT_TRAIL=DB, EXTENDED SCOPE=SPFILE;
```

5. Log in as user `jsmith` and then issue the same `ALTER SYSTEM` statement:

```
CONNECT jsmith
Enter password: password

ALTER SYSTEM SET AUDIT_TRAIL=OS, EXTENDED SCOPE=SPFILE;
```

The following output should appear:

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

Because user `jsmith` was assigned only the Confidential label, he cannot perform the `ALTER SYSTEM` statement.

6. Now log in as user `SYSTEM`, who normally has the `ALTER SYSTEM` privilege, and issue the same `ALTER SYSTEM` statement:

```
CONNECT SYSTEM
Enter password: password
```

The following output should appear:

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

`SYSTEM` no longer has sufficient privileges needed to perform an `ALTER SYSTEM` statement. Only users who have been assigned the Sensitive label, as with user `mdale`, can use the `ALTER SYSTEM` statement.

Step 6: Remove the Components for This Tutorial

1. Connect as the Oracle Label Security administrator and remove the label policy and its components.

```
CONNECT LBACSYS
Enter password: password

EXEC SA_SYSDBA.DROP_POLICY('PRIVACY', TRUE);
```

2. Connect as the Oracle Database Vault Owner and issue the following commands in the order shown, to set the ALTER SYSTEM command rule back to its previous setting and remove the rule set.

   For example:

```
CONNECT lbrown_dvowner
Enter password: password

EXEC DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE('ALTER SYSTEM', 'Allow System
Parameters','%', '%', 'Y');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('PII Rule Set', 'Check OLS
Factor');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Check OLS Factor');
EXEC DVSYS.DBMS_MACADM.DELETE_RULE_SET('PII Rule Set');
COMMIT;
```

3. Connect as the Database Vault Account Manager and remove users `mdale` and `jsmith`.

```
CONNECT amalcolm_dvacctmgr
Enter password: password

DROP USER mdale;
DROP USER jsmith;
```

Related Reports and Data Dictionary Views

Table 9–1 lists Oracle Database Vault reports that are useful for analyzing the integration of Oracle Database Vault and Oracle Label Security. See Chapter 16, "Oracle Database Vault Reports" for information about how to run these reports.

Table 9–1 Reports Related to Database Vault and Oracle Label Security Integration

Report	Description
"Factor Configuration Issues Report" on page 16-3	Lists factors in which the Oracle Label Security policy does not exist.
"Identity Configuration Issues Report" on page 16-3	Lists invalid label identities (the Oracle Label Security label for this identity has been removed and no longer exists).
"Security Policy Exemption Report" on page 16-10	Lists accounts and roles that have the `EXEMPT ACCESS POLICY` system privilege granted to them. Accounts that have this privilege can bypass all Virtual Private Database policy filters and any Oracle Label Security policies that use Oracle Virtual Private Database indirectly.

Table 9–2 lists data dictionary views that provide information about existing Oracle Label Security policies used with Oracle Database Vault.

Table 9–2 Data Dictionary Views Used for Oracle Label Security

Data Dictionary View	Description
"DBA_DV_MAC_POLICY View" on page 10-19	Lists the Oracle Label Security policies defined
"DBA_DV_MAC_POLICY_FACTOR View" on page 10-19	Lists the factors that are associated with Oracle Label Security policies

Table 9–2 (Cont.) Data Dictionary Views Used for Oracle Label Security

Data Dictionary View	Description
"DBA_DV_POLICY_LABEL View" on page 10-20	Lists the Oracle Label Security label for each factor identifier in the `DBA_DV_IDENTITY` view for each policy

Using Oracle Database Vault with Oracle Recovery Manager

You can use Recovery Manager (RMAN) in an Oracle Database Vault environment. The functionality of RMAN with Oracle Database Vault is the same as its functionality in a standard Oracle Database environment.

For more information about RMAN, see the following documentation:

- *Oracle Database Backup and Recovery Quick Start Guide*
- *Oracle Database Backup and Recovery Reference*

10

Oracle Database Vault Objects

This chapter contains:

- Oracle Database Vault Schemas
- Oracle Database Vault Roles
- Oracle Database Vault Accounts
- Oracle Database Vault Data Dictionary Views

Oracle Database Vault Schemas

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

Oracle Database Vault has the following schemas:

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

DVSYS Schema

The `DVSYS` schema contains Oracle Database Vault database objects, which store Oracle Database Vault configuration information and support the administration and run-time processing of Oracle Database Vault. In a default installation, the `DVSYS` schema is locked. The `DVSYS` schema also owns the `AUDIT_TRAIL$` table.

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 10-2 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 Account Manager Role, DV_ACCTMGR
- Oracle Database Vault PUBLIC Role, DV_PUBLIC
- Oracle Database Vault Security Analyst Role, DV_SECANALYST

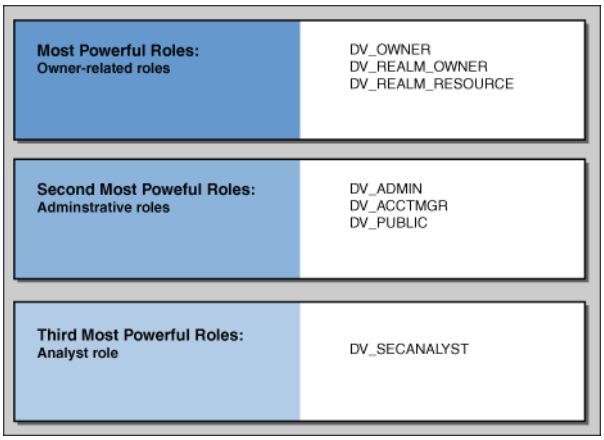
About Oracle Database Vault Roles

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. How you use these roles depends on the requirements that your company has in place.

Figure 10–1 illustrates the hierarchy of Oracle Database Vault roles, based on their levels of power.

Figure 10–1 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 10–1 summarizes the privileges available with Oracle Database Vault roles.

Table 10–1 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, EXEC	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]	No[2]
DVSYS schema, grant privileges on objects	No	No	No	No	No	No	No
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	No
Run Database Vault reports	Yes	No	No	Yes	No	Yes	No
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 10–1 (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 to manage the Oracle Database Vault roles and its configuration. In this guide, the example account that uses this role is `lbrown_dvowner`.

Privileges Associated with the DV_OWNER Role

The DV_OWNER role has the administrative capabilities that the DV_ADMIN role provides, and the reporting capabilities the DV_SECANALYST role provides. 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 addition to DV_ADMIN role, the DV_OWNER role has the GRANT ANY ROLE, ADMINISTER DATABASE TRIGGER, and ALTER ANY TRIGGER privileges.

> **Tip:** Consider creating a separate, named account for the DV_OWNER user. This way, if the user leaves the company, you can easily recreate this user account and then grant this user the DV_OWNER role.

To find the full list of system and object privileges associated with the DV_OWNER role, log in to SQL*Plus with administrative privileges and then enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_OWNER';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_OWNER';
```

When you install and register Oracle Database Vault, the DV_OWNER account is created. The user who is granted this role is also granted the ADMIN option and can run any Oracle Database Vault roles (except DV_ACCTMGR) without the ADMIN OPTION to any account. Users granted this role also can run Oracle Database Vault reports and monitor Oracle Database Vault.

How Are GRANT and REVOKE Operations Affected by the DV_OWNER Role?

Anyone with the DV_OWNER role can grant the DV_OWNER and DV_ADMIN roles to another user. The account granted this role can revoke any granted 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 right to grant or revoke the DV_OWNER role to or from any other database account. Note also that a user with the DV_OWNER role cannot grant or revoke the DV_ACCTMGR role.

What Happens When Oracle Database Vault Security Is Disabled?

The granting and revoking of all protected schema roles, including DV_OWNER, is 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 any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected schema roles.

Appendix B, "Disabling and Enabling Oracle Database Vault" explains 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. Grant this role to the database account owner who is responsible for managing one or more schema database accounts within a realm and the roles associated with the realm.

Privileges Associated with the DV_REALM_OWNER Role

A user who has been granted this role can use powerful system privileges like CREATE ANY, ALTER ANY, and DROP ANY within the realm. However, before this user can exercise these privileges, you must make this user either a participant or an owner for the realm. See "Defining Realm Authorization" on page 4-5 for instructions.

There are no object privileges granted to the DV_REALM_OWNER role, but it does have some system privileges. To find the full list of DV_REALM_OWNER system privileges, log in to SQL*Plus with administrative privileges and enter the following query:

```
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_REALM_OWNER';
```

How Are GRANT and REVOKE Operations Affected by the DV_REALM_OWNER Role?

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 must 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

Use the DV_REALM_RESOURCE role for operations such as creating tables, views, triggers, synonyms, and other objects that a realm would typically use.

Privileges Associated with the DV_REALM_RESOURCE Role

The DV_REALM_RESOURCE role provides the same system privileges as the Oracle RESOURCE role. In addition, both CREATE SYNONYM and CREATE VIEW are granted to this role.

There are no object privileges granted to the DV_REALM_RESOURCE role, but it does have some system privileges. To find the full list of DV_REALM_RESOURCE system privileges, log in to SQL*Plus with administrative privileges and enter the following query:

```
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_REALM_RESOURCE';
```

Though this role has system privilege grants that SYS controls, it does not have the DV_OWNER or DV_ADMIN role.

How Are GRANT and REVOKE Operations Affected by the DV_REALM_RESOURCE Role?

You can grant the DV_REALM_RESOURCE role to a database account that owns database tables, objects, triggers, views, procedures, and so on that are used to support any database application. This is a role designed for a schema type database account. The realm owner of the Oracle Data Dictionary realm, such as SYS, can grant this role to any database account or role.

Oracle Database Vault Configuration Administrator Role, DV_ADMIN

The DV_ADMIN role controls the Oracle Database Vault PL/SQL packages.

Privileges Associated with the DV_ADMIN Role

The DV_ADMIN role has the EXECUTE privilege on the DVSYS packages (DBMS_MACADM, DBMS_MACSECROLES, and DBMS_MACUTL). DV_ADMIN also has the capabilities provided by the DV_SECANALYST role, which allow the user to run Oracle Database Vault reports and monitor Oracle Database Vault. During installation, the DV_ADMIN role is granted to the DV_OWNER role with the ADMIN OPTION during installation.

To find the full list of system and object privileges associated with the DV_ADMIN role, log in to SQL*Plus with administrative privileges and then enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_ADMIN';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_ADMIN';
```

How Are GRANT and REVOKE Operations Affected by the DV_ADMIN Role?

Accounts such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone do not have the rights to grant or revoke DV_ADMIN from any other database account. The user with the DV_OWNER or DV_ADMIN role can grant or revoke this role to and from any database account.

What Happens When Oracle Database Vault Security Is Disabled?

The granting and revoking of protected schema roles, including DV_ADMIN, is 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 any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected schema roles.

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

Oracle Database Vault Account Manager Role, DV_ACCTMGR

Use the DV_ACCTMGR role to create and maintain database accounts and database profiles. In this manual, the example DV_ACCTMGR role is assigned to a user named amalcolm_dvacctmgr.

Privileges Associated with the DV_ACCTMGR Role

A user who has been granted this role can use the CREATE, ALTER, and DROP statements for users or profiles. However, a person who has been granted the DV_ACCTMGR role cannot perform the following operations:

- ALTER or DROP statements on the DVSYS account

- ALTER or DROP statements on users who have been granted the DV_ADMIN, DV_OWNER, and DV_SECANALYST roles
- Change passwords for users who have been granted the DV_ADMIN, DV_OWNER, and DV_SECANALYST roles

To find the full list of system and object privileges associated with the DV_ACCTMGR role, log in to SQL*Plus with administrative privileges and then enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_
ACCTMGR';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_ACCTMGR';
```

Tips:

- Oracle recommends that you add the user who has the DV_ACCTMGR role to the data dictionary realm. See "Step 1: Adding the SYSTEM User to the Data Dictionary Realm" on page 3-6 for an example.
- If you want the DV_ACCTMGR user to be able to grant other users ANY privileges, then log in as user SYS with the SYSDBA privilege and grant this user the GRANT ANY PRIVILEGE privilege.
- Consider creating a separate, named account for the DV_ACCTMGR user. This way, if this user forgets his or her password, you can log in as the original DV_ACCTMGR account when you recreate the user's password. Otherwise, you must disable Oracle Database Vault, log in as SYS or SYSTEM to recreate the password, and then re-enable Database Vault.

How Are GRANT and REVOKE Operations Affected by the DV_ACCTMGR Role?

Any account, such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone does not have the rights to grant this role to or revoke this role from any other database account. The account with the DV_ACCTMGR role and the ADMIN OPTION can grant this role without the ADMIN OPTION to any given database account and revoke this role from another account.

What Happens When Oracle Database Vault Security Is Disabled?

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 any account with the GRANT ANY ROLE privilege can perform GRANT and REVOKE operations on protected schema roles.

Appendix B, "Disabling and Enabling 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. (Remember that in a default installation, the DVSYS schema is locked.)

Privileges Associated with the DV_PUBLIC Role

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 14-1. 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 14-5. 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 Chapter 12, "Using the 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 Chapter 13, "Using the 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.

There are no system privileges granted to the DV_PUBLIC role, but it does have some object privileges. To find the full list of DV_PUBLIC object privileges, log in to SQL*Plus with administrative privileges and enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_
PUBLIC';
```

How Are GRANT and REVOKE Operations Affected by the DV_PUBLIC Role?

Oracle Database Vault does not enable you to directly grant object privileges in the DVSYS schema to PUBLIC. You must grant 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.

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 Data Dictionary Views" on page 10-11.

Privileges Associated with the DV_SECANALYST Role

There are no system privileges associated with the DV_SECANALYST role, but it does have the SELECT privilege for the DVSYS object schema and portions of the SYS and SYSMAN schema objects for reporting on DVSYS- and DVF-related entities. To find the full list of DV_SECANALYST object privileges, log in to SQL*Plus with administrative privileges and then enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_
SECANALYST';
```

How Are GRANT and REVOKE Operations Affected by the DV_SECANALYST Role?

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.

What Happens When Oracle Database Vault Security Is Disabled?

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, "Disabling and Enabling 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 choose not to create the Oracle Database Vault Account Manager account during installation, then both the DV_OWNER and DV_ACCTMGR roles are granted to the Oracle Database Vault Owner user account.

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

Table 10–2 lists the Oracle Database Vault database accounts that are needed in addition to the accounts that you create during installation.

Table 10–2 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

Table 10–2 (Cont.) Database Accounts Used by Oracle Database Vault

Database Account	Roles and Privileges	Description
LBACSYS	This account is created when you install Oracle Label Security by using the Oracle Universal Installer custom installation option. (It is not created when you install Oracle Database Vault.) Do not drop or re-create this account. If you plan to integrate a factor with an Oracle Label Security policy, you must assign this user as the owner of the realm that uses this factor. See "Using Oracle Database Vault Factors with Oracle Label Security Policies" on page 9-7 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 10–3 lists some model database accounts that can act as a guide. (The accounts listed in Table 10–3 serve as a guide to implementing Oracle Database Vault roles. These are not actual accounts that are created during installation.)

Table 10–3 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.

Table 10–3 (Cont.) Model Oracle Database Vault Database Accounts

Database Account	Roles and Privileges	Description
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.

Oracle Database Vault Data Dictionary Views

Oracle Database Vault provides a set of DBA-style data dictionary views that can be accessed through the DV_SECANALYST role or the DV_ADMIN role. (Alternatively, you can run reports on Oracle Database Vault. See Chapter 16, "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 administrative user to report on the state of the Oracle Database Vault configuration without having to perform the joins required to get the labels for codes that are stored in the core tables or from the related tables.

> **Note:** The DBA_REGISTRY data dictionary view, which displays information about the components loaded into a database, does not apply to Oracle Database Vault components. In other words, if you query this view, then no Oracle Database Vault components will be listed in the output.

This section contains:

- DBA_DV_CODE View
- DBA_DV_COMMAND_RULE View
- DBA_DV_FACTOR View
- DBA_DV_FACTOR_LINK View
- DBA_DV_FACTOR_TYPE View
- DBA_DV_IDENTITY View
- DBA_DV_IDENTITY_MAP View
- DBA_DV_MAC_POLICY View
- DBA_DV_MAC_POLICY_FACTOR View
- DBA_DV_POLICY_LABEL View
- DBA_DV_PUB_PRIVS View

- DBA_DV_REALM View
- DBA_DV_REALM_AUTH View
- DBA_DV_REALM_OBJECT View
- DBA_DV_ROLE View
- DBA_DV_RULE View
- DBA_DV_RULE_SET View
- DBA_DV_RULE_SET_RULE View
- DBA_DV_USER_PRIVS View
- DBA_DV_USER_PRIVS_ALL View

DBA_DV_CODE View

The DBA_DV_CODE data dictionary view lists generic lookup codes for the user interface, error messages, constraint checking, and so on. These codes are used for the user interface, views, and for validating input in a translatable fashion.

For example:

```
SELECT CODE, VALUE FROM DVSYS.DBA_DV_CODE WHERE CODE_GROUP = 'BOOLEAN';
```

Output similar to the following appears:

```
CODE    VALUE
------- --------
Y       True
N       False
```

Column	Datatype	Null	Description
CODE_GROUP	VARCHAR(30)	NOT NULL	Displays one of the following code groups: ■ AUDIT_EVENTS: Contains the action numbers and action names that are used for the custom event audit trail records ■ BOOLEAN: A simple Yes or No or True or False lookup ■ DB_OBJECT_TYPE: The database object types that can be used for realm objects and command authorizations ■ SQL_CMDS: The DDL commands that can be protected through command rules ■ FACTOR_AUDIT: The auditing options for factor retrieval processing ■ FACTOR_EVALUATE: The evaluation options (by session or by access) for factor retrieval ■ FACTOR_FAIL: The options for propagating errors when a factor retrieval method fails ■ FACTOR_IDENTIFY: The options for determining how a factor identifier is resolved (for example, by method or by factors) ■ FACTOR_LABEL: The options for determining how a factor identifier is labeled in the session establishment phase ■ LABEL_ALG: The algorithms that can be used to determine the maximum session label for a database session for each policy. See Table 11–57, " Oracle Label Security Merge Algorithm Codes" on page 11-49 for a listing of the Oracle Label Security merge algorithm codes. ■ OPERATORS: The Boolean operators that can be used for identity maps ■ REALM_AUDIT: The options for auditing realm access or realm violations ■ REALM_OPTION: The options for ownership of a realm ■ RULESET_AUDIT: The options for auditing rule set execution or rule set errors ■ RULESET_EVALUATE: The options for determining the success or failure of a rule set based on all associated rules being true or any associated rule being true ■ RULESET_EVENT: The options to invoke a custom event handler when a rule set evaluates to Succeeds or Fails ■ RULESET_FAIL: The options to determine the run-time visibility of a rule set failing
CODE	VARCHAR(30)	NOT NULL	Boolean code used; either Y (yes) or N (no).
VALUE	VARCHAR(4000)		Boolean value used; either True if the Boolean code is Y or False if the Boolean code is N.

Column	Datatype	Null	Description
LANGUAGE	VARCHAR(3)	NOT NULL	Language for this installation of Oracle Database Vault. Supported languages are as follows: ■ en: English ■ de: German ■ es: Spanish ■ fr: French ■ it: Italian ■ ja: Japanese ■ ko: Korean ■ pt_BR: Brazilian Portuguese ■ zh_CN: Simplified Chinese ■ zh_TW: Traditional Chinese
DESCRIPTION	VARCHAR(1024)		Brief description of the code group.

DBA_DV_COMMAND_RULE View

The DBA_DV_COMMAND_RULE data dictionary view lists the SQL statements that are protected by command rules. See Chapter 6, "Configuring Command Rules" for more information about command rules.

For example:

```
SELECT COMMAND, RULE_SET_NAME FROM DVSYS.DBA_DV_COMMAND_RULE;
```

Output similar to the following appears:

```
COMMAND         RULE_SET_NAME
--------------- ----------------------------
GRANT           Can Grant VPD Administration
REVOKE          Can Grant VPD Administration
ALTER SYSTEM    Allow System Parameters
ALTER USER      Can Maintain Own Account
CREATE USER     Can Maintain Account/Profiles
DROP USER       Can Maintain Account/Profiles
CREATE PROFILE  Can Maintain Account/Profiles
DROP PROFILE    Can Maintain Account/Profiles
ALTER PROFILE   Can Maintain Account/Profiles
```

Column	Datatype	Null	Description
COMMAND	VARCHAR(30)	NOT NULL	Name of the command rule. For a list of default command rules, see Default Command Rules on page 6-2.
RULE_SET_NAME	VARCHAR(90)	NOT NULL	Name of the rule set associated with this command rule.
OBJECT_OWNER	VARCHAR(30)	NOT NULL	The owner of the object that the command rule affects.
OBJECT_NAME	VARCHAR(128)	NOT NULL	The name of the database object the command rule affects, (for example, a database table).
ENABLED	VARCHAR(1)	NOT NULL	Y indicates the command rule is enabled; N indicates it is disabled.

DBA_DV_FACTOR View

The DBA_DV_FACTOR data dictionary view lists the existing factors in the current database instance.

For example:

```
SELECT NAME, GET_EXPR FROM DVSYS.DBA_DV_FACTOR WHERE NAME = 'Session_User';
```

Output similar to the following appears:

```
NAME          GET_EXPR
------------- ---------------------------------------------
Session_User  UPPER(SYS_CONTEXT('USERENV', 'SESSION_USER'))
```

Related Views

- DBA_DV_FACTOR_LINK View
- DBA_DV_FACTOR_TYPE View

Column	Datatype	Null	Description
NAME	VARCHAR2(30)	NOT NULL	Name of the factor. See "Default Factors" on page 7-2 for a list of default factors.
DESCRIPTION	VARCHAR2(4000)		Description of the factor.
FACTOR_TYPE_NAME	VARCHAR2(90)	NOT NULL	Category of the factor, which is used to classify the purpose of the factor.
ASSIGN_RULE_SET_NAME	VARCHAR2(90)		Rule set used to control the identify of the factor.
GET_EXPR	VARCHAR2(1024)		PL/SQL expression that retrieves the identity of a factor.
VALIDATE_EXPR	VARCHAR2(1024)		PL/SQL expression used to validate the identify of the factor. It returns a Boolean value.
IDENTIFIED_BY	NUMBER	NOT NULL	Determines the identity of a factor, based on the expression listed in the GET_EXPR column. Possible values are: ■ 0: By constant ■ 1: By method ■ 2: By factors
IDENTIFIED_BY_MEANING	VARCHAR2(4000)		Provides a text description for the corresponding value in the IDENTIFIED_BY column. Possible values are: ■ By Constant: If IDENTIFIED_COLUMN is 0 ■ By Method: If IDENTIFIED_COLUMN is 1 ■ By Factors: If IDENTIFIED_COLUMN is 2
LABELED_BY	NUMBER	NOT NULL	Determines the labeling the factor: ■ 0: Labels the identities for the factor directly from the labels associated with an Oracle Label Security policy ■ 1: Derives the factor identity label from the labels of its child factor identities.

Column	Datatype	Null	Description
LABELED_BY_MEANING	VARCHAR2(4000)		Provides a text description for the corresponding value in the LABELED_BY column. Possible values are: ■ By Self: If LABELED_BY column is 0 ■ By Factors: If LABELED_BY column is 1
EVAL_OPTIONS	NUMBER	NOT NULL	Determines how the factor is evaluated when the user logs on: ■ 0: When the database session is created ■ 1: Each time the factor is accessed ■ 2: On start-up
EVAL_OPTIONS_MEANING	VARCHAR2(4000)		Provides a text description for the corresponding value in the EVAL_OPTIONS column. Possible values are: ■ For Session: If EVAL_OPTIONS is 0 ■ By Access: If EVAL_OPTIONS is 1 ■ On Startup: If EVAL_OPTIONS is 2
AUDIT_OPTIONS	NUMBER	NOT NULL	Option for auditing the factor if you want to generate a custom Oracle Database Vault audit record. Possible values are: ■ 0: No auditing set ■ 1: Always audits ■ 2: Audits if get_expr returns an error ■ 4: Audits if get_expr is null ■ 8: Audits if the validation procedure returns an error ■ 16: Audits if the validation procedure is false ■ 32: Audits if there is no trust level set ■ 64: Audits if the trust level is negative.
FAIL_OPTIONS	NUMBER	NOT NULL	Options for reporting factor errors: ■ 1: Shows an error message. ■ 2: Does not show an error message.
FAIL_OPTIONS_MEANING	VARCHAR2(4000)		Provides a text description for the corresponding value in the FAIL_OPTIONS column. Possible values are: ■ Show Error Message ■ Do Not Show Error Message:

DBA_DV_FACTOR_LINK View

The DBA_DV_FACTOR_LINK data dictionary view shows the relationships of each factor whose identity is determined by the association of child factors. 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.

For example:

```
SELECT PARENT_FACTOR_NAME, CHILD_FACTOR_NAME FROM DVSYS.DBA_DV_FACTOR_LINK;
```

Output similar to the following appears:

```
PARENT_FACTOR_NAME             CHILD_FACTOR_NAME
------------------------------ ------------------------------
Domain                         Database_Instance
Domain                         Database_IP
Domain                         Database_Hostname
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_FACTOR_TYPE View

Column	Datatype	Null	Description
PARENT_FACTOR_NAME	VARCHAR(30)	NOT NULL	Name of the parent factor.
CHILD_FACTOR_NAME	VARCHAR(30)	NOT NULL	Name of the child factor of the parent factor.
LABEL_IND	VARCHAR(1)	NOT NULL	Indicates whether the child factor that is linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Possible values are: ■ Y (for Yes) ■ N (for No)

DBA_DV_FACTOR_TYPE View

The DBA_DV_FACTOR_TYPE data dictionary view lists the names and descriptions of factor types used in the system.

For example:

```
SELECT * FROM DVSYS.DBA_DV_FACTOR_TYPE WHERE NAME = 'Hostname';
```

Output similar to the following appears:

```
NAME      DESCRIPTION
--------- --------------------------------------------------------------------------
Time      Time-based factor
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_FACTOR_LINK View

Column	Datatype	Null	Description
NAME	VARCHAR(90)	NOT NULL	Name of the factor type.
DESCRIPTION	VARCHAR(1024)		Description of the factor type.

DBA_DV_IDENTITY View

The DBA_DV_IDENTITY data dictionary view lists the identities for each factor.

For example:

```
SELECT * FROM DVSYS.DBA_DV_IDENTITY WHERE VALUE = 'GLOBAL SHARED';
```

Output similar to the following appears, assuming you have created only one factor identity:

```
FACTOR_NAME          VALUE          TRUST_LEVEL
---------------      -------------- ------------
Identification_Type  GLOBAL SHARED  1
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_IDENTITY_MAP View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(30)	NOT NULL	Name of the factor.
VALUE	VARCHAR(1024)	NOT NULL	Value of the factor.
TRUST_LEVEL	NUMBER	NOT NULL	Number that indicates the magnitude of trust relative to other identities for the same factor.

DBA_DV_IDENTITY_MAP View

The DBA_DV_IDENTITY_MAP data dictionary view lists the mappings for each factor identity. The view includes mapping factors that are identified by other factors to combinations of parent-child factor links. For each factor, the maps are joined by the OR operation, and for different factors, the maps are joined by the AND operation.

You can use this view to resolve the identity for factors that are identified by other factors (for example, a domain) or for factors that have continuous domains (for example, Age or Temperature).

For example:

```
SELECT FACTOR_NAME, IDENTITY_VALUE FROM DVSYS.DBA_DV_IDENTITY_MAP;
```

Output similar to the following appears:

```
FACTOR_NAME      IDENTITY_VALUE
---------------- --------------------
Sector2_Program  Accounting-Sensitive
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_IDENTITY View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(30)	NOT NULL	Factor the identity map is for.
IDENTITY_VALUE	VARCHAR(1024)	NOT NULL	Value the factor assumes if the identity map evaluates to TRUE.
OPERATION_VALUE	VARCHAR(4000)		Relational operator for the identity map (for example, <, >, =, and so on)
OPERAND1	VARCHAR(1024)		Left operand for the relational operator; refers to the low value you enter.
OPERAND2	VARCHAR(1024)		Right operand for the relational operator; refers to the high value you enter.
PARENT_FACTOR_NAME	VARCHAR(30)		The parent factor link to which the map is related.
CHILD_FACTOR_NAME	VARCHAR(30)		The child factor link to which the map is related.

Column	Datatype	Null	Description
LABEL_IND	VARCHAR(1)		Indicates whether the child factor being linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Possible values are: ■ Y (for Yes) ■ N (for No)

DBA_DV_MAC_POLICY View

The DBA_DV_MAC_POLICY data dictionary view lists the Oracle Label Security policies defined for use with Oracle Database Vault.

For example:

```
SELECT POLICY_NAME, ALGORITHM_CODE, ALGORITHM_MEANING
 FROM DVSYS.DBA_DV_MAC_POLICY;
```

Output similar to the following appears:

```
POLICY_NAME     ALGORITHM_CODE    ALGORITHM_MEANING
--------------- ----------------- ----------------------
ACCESS_DATA     LUI               Minimum Level/Union/Intersection
```

Related Views

- DBA_DV_MAC_POLICY_FACTOR View
- DBA_DV_POLICY_LABEL View

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(30)	NOT NULL	Name of the policy.
ALGORITHM_CODE	VARCHAR(30)	NOT NULL	Merge algorithm code used for the policy. See Table 11–57 on page 11-49 for a listing of algorithm codes.
ALGORITHM_MEANING	VARCHAR(4000)		Provides a text description for the corresponding value in the ALGORITHM_CODE column. See Table 11–57 on page 11-49 for a listing of algorithm code descriptions.
ERROR_LABEL	VARCHAR(4000)		Label specified for initialization errors, to be set when a configuration error or run-time error occurs during session initialization.

DBA_DV_MAC_POLICY_FACTOR View

The DBA_DV_MAC_POLICY data dictionary view lists the factors that are associated with Oracle Label Security policies.

You can use this view to determine what factors contribute to the maximum session label for each policy using the DBA_DV_MAC_POLICY view.

For example:

```
SELECT * FROM DVSYS.DBA_DV_MAC_POLICY_FACTOR;
```

Output similar to the following appears:

```
FACTOR_NAME    MAC_POLICY_NAME
-------------- ------------------
App_Host_Name  Access Locations
```

Related Views

- DBA_DV_MAC_POLICY View
- DBA_DV_POLICY_LABEL View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(30)	NOT NULL	Name of the factor.
MAC_POLICY_NAME	VARCHAR(30)	NOT NULL	Name of the Oracle Label Security policy associated with this factor.

DBA_DV_POLICY_LABEL View

The DBA_DV_POLICY_LABEL data dictionary view lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.

For example:

```
SELECT * FROM DVSYS.DBA_DV_POLICY_LABEL;
```

Output similar to the following appears:

```
IDENTITY_VALUE   FACTOR_NAME     POLICY_NAME       LABEL
---------------- --------------  ----------------  ---------
App_Host_Name    Sect2_Fin_Apps  Access Locations  Sensitive
```

Related Views

- DBA_DV_MAC_POLICY View
- DBA_DV_MAC_POLICY_FACTOR View

Column	Datatype	Null	Description
IDENTITY_VALUE	VARCHAR(1024)	NOT NULL	Name of the factor identifier.
FACTOR_NAME	VARCHAR(30)	NOT NULL	Name of the factor associated with the factor identifier.
POLICY_NAME	VARCHAR(30)	NOT NULL	Name of the Oracle Label Security policy associated with this factor.
LABEL	VARCHAR(4000)	NOT NULL	Name of the Oracle Label Security label associated with the policy.

DBA_DV_PUB_PRIVS View

The DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in the Oracle Database Vault Administrator (DV_ADMIN). See also "Privilege Management - Summary Reports" on page 16-9.

For example:

```
SELECT USERNAME, ACCESS_TYPE FROM DVSYS.DBA_DV_PUB_PRIVS WHERE USERNAME = 'OE';
```

Output similar to the following appears:

```
USERNAME    ACCESS_TYPE
----------- -----------------
OE          PUBLIC
```

Related Views

- DBA_DV_USER_PRIVS View

- DBA_DV_USER_PRIVS_ALL View
- DBA_DV_ROLE View

Column	Datatype	Null	Description
USERNAME	VARCHAR(30)	NOT NULL	Database schema in the current database instance.acces
ACCESS_TYPE	VARCHAR(30)		Access type granted to the user listed in the USERNAME column (for example, PUBLIC).
PRIVILEGE	VARCHAR(40)	NOT NULL	Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(30)	NOT NULL	Owner of the database schema to which the USERNAME user has been granted privileges.
OBJECT_NAME	VARCHAR(30)	NOT NULL	Name of the object within the schema listed in the OWNER column.

DBA_DV_REALM View

The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.

For example:

```
SELECT NAME, AUDIT_OPTIONS, ENABLED FROM DVSYS.DBA_DV_REALM
  WHERE AUDIT_OPTIONS = 'N';
```

Output similar to the following appears:

```
NAME                           AUDIT_OPTIONS    ENABLED
------------------------------ ---------------- --------
Performance Statistics Realm   N                1
```

Related Views

- DBA_DV_REALM_AUTH View
- DBA_DV_REALM_OBJECT View

Column	Datatype	Null	Description
NAME	VARCHAR(90)	NOT NULL	Names of the realms created. See"Default Realms" on page 4-2 for a listing of default realms.
DESCRIPTION	VARCHAR(1024)	NOT NULL	Description of the realm created.
AUDIT_OPTIONS	NUMBER	NOT NULL	Specifies whether auditing is enabled. Possible values are: ■ 0: No auditing for the realm. ■ 1: Creates an audit record when a realm violation occurs (for example, when an unauthorized user tries to modify an object that is protected by the realm). ■ 2: Creates an audit record for authorized activities on objects protected by the realm. ■ 3: Creates an audit record for both authorized and unauthorized activities on objects protected by the realm.
ENABLED	VARCHAR(1)	NOT NULL	Specifies whether realm checking is enabled. Y (yes) indicates it is enabled; N (no) indicates it is not.

DBA_DV_REALM_AUTH View

The DBA_DV_REALM_AUTH data dictionary view lists the authorization of a named database user account or database role (GRANTEE) to access realm objects in a particular realm. See Defining Realm Authorization on page 4-5 for more information.

For example:

```
SELECT REALM_NAME, GRANTEE, AUTH_RULE_SET_NAME FROM DVSYS.DBA_DV_REALM_AUTH;
```

Output similar to the following appears:

```
REALM_NAME                    GRANTEE  AUTH_RULE_SET_NAME
----------------------------- -------- ---------------------
Performance Statistics Realm  SYSADM   Check Conf Access
```

Related Views

- DBA_DV_REALM View
- DBA_DV_REALM_OBJECT View

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(90)	NOT NULL	Name of the realm.
GRANTEE	VARCHAR(30)	NOT NULL	User or role name to authorize as owner or participant.
AUTH_RULE_SET_NAME	VARCHAR(90)		Rule set to check before authorizing. If the rule set evaluates to TRUE, then the authorization is allowed.
AUTH_OPTIONS	VARCHAR(4000)		Type of realm authorization: either Participant or Owner.

DBA_DV_REALM_OBJECT View

The DBA_DV_REALM_OBJECT data dictionary view lists the database schemas, or subsets of schemas with specific database objects contained therein, that are secured by the realms. See "Creating Realm-Secured Objects" on page 4-4 for more information.

For example:

```
SELECT REALM_NAME, OWNER, OBJECT_NAME FROM DVSYS.DBA_DV_REALM_OBJECT;
```

Output similar to the following appears:

```
REALM_NAME                   OWNER    OBJECT_NAME
---------------------------- -------- -----------
Performance Statistics Realm OE       ORDERS
```

Related Views

- DBA_DV_REALM View
- DBA_DV_REALM_AUTH View

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(90)	NOT NULL	Name of the realm.
OWNER	VARCHAR(90)	NOT NULL	Database schema owner who owns the realm.
OBJECT_NAME	VARCHAR(90)	NOT NULL	Name of the object the realm protects.

Column	Datatype	Null	Description
OBJECT_TYPE	VARCHAR(90)	NOT NULL	Type of object the realm protects, such as a database table, view, index, or role.

DBA_DV_ROLE View

The DBA_DV_ROLE data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.

For example:

```
SELECT ROLE, RULE_NAME FROM DVSYS.DBA_DV_ROLE;
```

Output similar to the following appears:

```
ROLE                 RULE_NAME
-------------------- --------------------
Sector2_APP_MGR      Check App2 Access
Sector2_APP_DBA      Check App2 Access
```

Related Views

- DBA_DV_PUB_PRIVS View
- DBA_DV_USER_PRIVS View
- DBA_DV_USER_PRIVS_ALL View

Column	Datatype	Null	Description
ROLE	VARCHAR(30)	NOT NULL	Name of the secure application role.
RULE_NAME	VARCHAR(90)	NOT NULL	Name of the rule set associated with the secure application role.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the secure application role is enabled. Y (yes) enables the role; N (no) disables it.

DBA_DV_RULE View

The DBA_DV_RULE data dictionary view lists the rules that have been defined.

For example:

```
SELECT * FROM DVSYS.DBA_DV_RULE WHERE NAME = 'Maintenance Window';
```

Output similar to the following appears:

```
NAME                 RULE_EXP
-------------------- ------------------------------------------------
Maintenance Window   TO_CHAR(SYSDATE,'HH24') BETWEEN '10' AND '12'
```

To find the rule sets that use specific rules, query the DBA_DV_RULE_SET_RULE view.

Related Views

- DBA_DV_RULE_SET View
- DBA_DV_RULE_SET_RULE View

Column	Datatype	Null	Description
NAME	VARCHAR(90)	NOT NULL	Name of the rule.

Column	Datatype	Null	Description
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression for the rule.

DBA_DV_RULE_SET View

The DBA_DV_RULE_SET data dictionary view lists the rules sets that have been created.

For example:

```
SELECT RULE_SET_NAME, HANDLER_OPTIONS, HANDLER FROM DVSYS.DBA_DV_RULE_SET
 WHERE RULE_SET_NAME = 'Maintenance Period';
```

Output similar to the following appears:

```
RULE_SET_NAME       HANDLER_OPTIONS  HANDLER
------------------- ---------------- ----------------------
Maintenance Period                   1 dbavowner.email_alert
```

Related Views

- DBA_DV_RULE View
- DBA_DV_RULE_SET_RULE View

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(90)	NOT NULL	Name of the rule set.
DESCRIPTION	VARCHAR(1024)		Description of the rule set.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the rule set has been enabled. Y (yes) enables the rule set; N (no) disables it.
EVAL_OPTIONS_MEANING	VARCHAR(4000)		For rules sets that contain multiple rules, determines how many rules are evaluated. Possible values are: ■ All True: All rules in the rule set must evaluate to true for the rule set itself to evaluate to TRUE. ■ Any True: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to TRUE.
AUDIT_OPTIONS	NUMBER	NOT NULL	Indicates when auditing is used. Possible values are: ■ 0: No auditing ■ 1: Audit on failure ■ 2: Audit on success ■ 3: Audit on both failure and success
FAIL_OPTIONS_MEANING	VARCHAR(4000)		Determines when an audit record is created for the rule set. Possible values are: ■ Do Not Show Error Message. ■ Show Error Message
FAIL_MESSAGE	VARCHAR(80)		Error message for failure that is associated with the fail code listed in the FAIL_CODE column.

Column	Datatype	Null	Description
FAIL_CODE	VARCHAR(10)		The error message number associated with the message listed in the FAIL_MESSAGE column. Possible values are in the range of negative -20000 to -20999.
HANDLER_OPTIONS	NUMBER	NOT NULL	Determines how error handling is used. Possible values are: ■ 0: Disables error handling. ■ 1: Call handler on rule set failure. ■ 2: Call handler on rule set success.
HANDLER	VARCHAR(1024)		Name of the PL/SQL function or procedure that defines the custom event handler logic.

DBA_DV_RULE_SET_RULE View

The DBA_DV_RULE_SET_RULE data dictionary view lists rules that are associated with existing rule sets.

For example:

```
SELECT RULE_SET_NAME, RULE_NAME, RULE_EXPR FROM DVSYS.DBA_DV_RULE_SET_RULE
 WHERE RULE_NAME = 'Is Security Officer';
```

Output similar to the following appears:

```
RULE_SET_NAME                RULE_NAME          RULE_EXP
---------------------------- ------------------ --------------------------------
Can Grant VPD Administration Is Security Owner  DVSYS.DBMS_MACUTL.USER_HAS_ROLE_
                                                VARCHAR('DV_OWNER',dvsys.dv_
                                                login_user) = 'Y'
```

Related Views

- DBA_DV_RULE View
- DBA_DV_RULE_SET View

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(90)	NOT NULL	Name of the rule set that contains the rule.
RULE_NAME	VARCHAR(90)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression that defines the rule listed in the RULE_NAME column.
ENABLED	VARCHAR(1)		Indicates whether the rule is enabled or disabled. Y (yes) enables the rule set; N (no) disables it.
RULE_ORDER	NUMBER	NOT NULL	The order in which rules are used within the rule set. Does not apply to this release.

DBA_DV_USER_PRIVS View

The DBA_DV_USER_PRIVS data dictionary view lists the privileges for a database user account excluding privileges granted through the PUBLIC role.

For example:

```
SELECT USERNAME, ACCESS_TYPE, PRIVILEGE FROM DVSYS.DBA_DV_USER_PRIVS;
```

Output similar to the following appears:

```
USERNAME  ACCESS_TYPE          PRIVILEGE
--------- -------------------- ------------
DVSYS     DV_PUBLIC            EXECUTE
DVOWNER   DV_ADMIN             SELECT
SYS       SELECT_CATALOG_ROLE  SELECT
...
```

Related Views

- DBA_DV_PUB_PRIVS View
- DBA_DV_ROLE View
- DBA_DV_USER_PRIVS_ALL View

Column	Datatype	Null	Description
USERNAME	VARCHAR(30)	NOT NULL	Name of the database schema account in which privileges have been defined.
ACCESS_TYPE	VARCHAR(30)		Role the database user account listed in the USERNAME column uses to access the database. Oracle Database Vault accounts have direct access.
PRIVILEGE	VARCHAR(40)	NOT NULL	Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(30)	NOT NULL	Name of the database user account.
OBJECT_NAME	VARCHAR(30)	NOT NULL	Name of the PL/SQL function or procedure used to define privileges.

DBA_DV_USER_PRIVS_ALL View

The DBA_DV_USER_PRIVS_ALL data dictionary view lists the privileges for a database account including privileges granted through PUBLIC.

For example:

```
SELECT USERNAME, ACCESS_TYPE, PRIVILEGE FROM DVSYS.DBA_DV_USER_PRIVS;
```

Output similar to the following appears:

```
USERNAME            ACCESS_TYPE  PRIVILEGE
------------------- ------------ -----------------
AMALCOLM_DVACCTMGR  CONNECT      CREATE_SESSION
LBROWN_DVOWNER      DIRECT       CREATE PROCEDURE
...
```

Related Views

- DBA_DV_PUB_PRIVS View
- DBA_DV_ROLE View
- DBA_DV_USER_PRIVS View

Column	Datatype	Null	Description
USERNAME	VARCHAR(30)		Name of the database schema account in which privileges have been defined.

Column	Datatype	Null	Description
ACCESS_TYPE	VARCHAR(30)		Role the database user account listed in the USERNAME column uses to access the database. Oracle Database Vault accounts have direct access.
PRIVILEGE	VARCHAR(40)		Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(30)		Name of the database user account.
OBJECT_NAME	VARCHAR(30)		Name of the PL/SQL function or procedure used to define privileges.

11

Using the DVSYS.DBMS_MACADM Package

This chapter contains:

- About the DVSYS.DBMS_MACADM Package
- Realm Procedures Within DVSYS.DBMS_MACADM
- Rule Set Procedures Within DVSYS.DBMS_MACADM
- Command Rule Procedures Within DVSYS.DBMS_MACADM
- Factor Procedures and Functions Within DVSYS.DBMS_MACADM
- Secure Application Role Procedures Within DVSYS.DBMS_MACADM
- Oracle Label Security Policy Procedures Within DVSYS.DBMS_MACADM

About the DVSYS.DBMS_MACADM Package

The procedures and 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.

Many of the parameters used in the procedures and functions in the `DVSYS.DBMS_MACADM` package can use the constants available in the `DVSYS.DBMS_MACUTL` package. See "DVSYS.DBMS_MACUTL Constants" on page 13-1 for more information.

Oracle Database Vault also provides a set of PL/SQL interfaces that you can use with the `DVSYS.DBMS_MACADM` package. See Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces," for more information.

Realm Procedures Within DVSYS.DBMS_MACADM

Table 11–1 lists procedures within the `DVSYS.DBMS_MACADM` package that you can use to configure realms. For constants that you can use with these procedures, see Table 13–1 on page 13-1 for more information.

Chapter 4, "Configuring Realms" describes realms in detail. See also Chapter 13, "Using the DVSYS.DBMS_MACUTL Package" for a set of general purpose utility procedures that you can use with the realm procedures.

Table 11–1 DVSYS.DBMS_MACADM Realm Configuration Procedures

Procedure	Description
ADD_AUTH_TO_REALM Procedure	Authorizes a user or role to access a realm as an owner or a participant
ADD_OBJECT_TO_REALM Procedure	Registers a set of objects for realm protection
CREATE_REALM Procedure	Creates a realm
DELETE_AUTH_FROM_REALM Procedure	Removes the authorization of a user or role to access a realm
DELETE_OBJECT_FROM_REALM Procedure	Removes a set of objects from realm protection
DELETE_REALM Procedure	Deletes a realm
DELETE_REALM_CASCADE Procedure	Deletes a realm, including its related Database Vault configuration information
RENAME_REALM Procedure	Renames a realm. The name change takes effect everywhere the realm is used
UPDATE_REALM Procedure	Updates a realm
UPDATE_REALM_AUTH Procedure	Updates the authorization of a user or role to access a realm

ADD_AUTH_TO_REALM Procedure

The `ADD_AUTH_TO_REALM` procedure authorizes a user or role to access a realm as an owner or a participant. The person running this procedure cannot add himself or herself to the realm as a realm participant. For detailed information about realm authorization, see "Defining Realm Authorization" on page 4-5.

Optionally, you can specify a rule set that must be checked before allowing the authorization to proceed, a rule set.

Syntax

```
DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    IN VARCHAR2,
  grantee       IN VARCHAR2,
  rule_set_name IN VARCHAR2,
  auth_options  IN NUMBER);
```

Parameters

Table 11–2 ADD_AUTH_TO_REALM Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DVSYS.DBA_DV_REALM` view, described in "DBA_DV_REALM View" on page 10-21.

Table 11–2 (Cont.) ADD_AUTH_TO_REALM Parameters

Parameter	Description
grantee	User or role name to authorize as an owner or a participant. You cannot select yourself (that is, the user logging in) or anyone who has been granted the DV_OWNER, DV_ADMIN, or DV_SECANALYST roles. To find the existing users and roles in the current database instance, query the DBA_USERS and DBA_ROLES views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view, described in "DBA_DV_REALM_AUTH View" on page 10-22. To find existing secure application roles used in privilege management, query the DVSYS.DBA_DV_ROLE view. Both are described in "Oracle Database Vault Data Dictionary Views" on page 10-11.
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, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24. To find rules that are associated with the rule sets, query the DVSYS.DBA_DV_RULE_SET_RULE view, described in "DBA_DV_RULE_SET_RULE View" on page 10-25.
auth_options	Optional. Specify one of the following options to authorize the realm: ■ 0: 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. ■ 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 multiple owners. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_REALM_AUTH_PARTICIPANT (default) ■ G_REALM_AUTH_OWNER See "Defining Realm Authorization" on page 4-5 for more information on participants and owners.

Examples

The following example authorizes user SYSADM as a participant in the Performance Statistics Realm.

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name  => 'Performance Statistics Realm',
  grantee     => 'SYSADM');
END;
/
```

This example sets user SYSADM as the owner of the Performance Statistics Realm.

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name   => 'Performance Statistics Realm',
  grantee      => 'SYSADM',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

The next example triggers the Check Conf Access rule set before allowing user SYSADM to be made the owner of the Performance Statistics Realm.

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    => 'Performance Statistics Realm',
  grantee       => 'SYSADM',
  rule_set_name => 'Check Conf Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

ADD_OBJECT_TO_REALM Procedure

This procedure registers a set of objects for realm protection.

Syntax

```
DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   IN VARCHAR2,
  object_owner IN VARCHAR2,
  object_name  IN VARCHAR2,
  object_type  IN VARCHAR2);
```

Parameters

Table 11–3 ADD_OBJECT_TO_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21
object_owner	The owner of the object that is being added to the realm. If you add a role to a realm, the object owner of the role is shown as % (for all), because roles do not have owners. To find the available users, query the DBA_USERS view, described in *Oracle Database Reference*. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view, described in "DBA_DV_REALM_AUTH View" on page 10-22.
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 %.) You can also use the DVSYS.DBMS_MACUTL G_ALL_OBJECT constant. To find the available objects, query the ALL_OBJECTS view, described in *Oracle Database Reference*. To find objects that are secured by existing realms, query the DVSYS.DBA_DV_REALM_OBJECT view, described in "DBA_DV_REALM_OBJECT View" on page 10-22.
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 %.) You can also use the DVSYS.DBMS_MACUTL G_ALL_OBJECT constant.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'Performance Statistics Realm',
  object_owner => '%',
  object_name  => 'GATHER_SYSTEM_STATISTICS',
  object_type  => 'ROLE');
END;
/
```

CREATE_REALM Procedure

The CREATE_REALM procedure creates a realm. After you create the realm, use the following procedures to complete the realm definition:

- ADD_OBJECT_TO_REALM procedure registers one or more objects for the realm.
- ADD_AUTH_TO_REALM procedures authorize users or roles for the realm.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_REALM(
  realm_name    IN VARCHAR2,
  description   IN VARCHAR2,
  enabled       IN VARCHAR2,
  audit_options IN NUMBER);
```

Parameters

Table 11–4 CREATE_REALM Parameters

Parameter	Description
realm_name	Realm name, up to 90 characters in mixed-case. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21.
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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_YES ■ G_NO

Table 11–4 (Cont.) CREATE_REALM Parameters

Parameter	Description
audit_options	Specify one of the following options 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 You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_REALM_AUDIT_OFF ■ G_REALM_AUDIT_SUCCESS ■ G_REALM_AUDIT_FAIL ■ G_REALM_AUDIT_FAIL + G_REALM_AUDIT_SUCCESS

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Performance Statistics Realm',
  description   => 'Realm to measure performance',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL);
END;
/
```

See Also: Example 13–1, "Creating a Realm Using DVSYS.DBMS_MACUTL Constants" on page 13-4

DELETE_AUTH_FROM_REALM Procedure

The DELETE_AUTH_FROM_REALM procedure removes the authorization of a user or role to access a realm.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_AUTH_FROM_REALM(
  realm_name IN VARCHAR2,
  grantee    IN VARCHAR2);
```

Parameters

Table 11–5 DELETE_AUTH_FROM_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21
grantee	User or role name. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view, described in "DBA_DV_REALM_AUTH View" on page 10-22.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_AUTH_FROM_REALM(
  realm_name => 'Performance Statistics Realm',
  grantee    => 'SYS');
END;
/
```

DELETE_OBJECT_FROM_REALM Procedure

The DELETE_OBJECT_FROM_REALM procedure removes a set of objects from realm protection.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_OBJECT_FROM_REALM(
  realm_name   IN VARCHAR2,
  object_owner IN VARCHAR2,
  object_name  IN VARCHAR2,
  object_type  IN VARCHAR2);
```

Parameters

Table 11–6 DELETE_OBJECT_FROM_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21
object_owner	The owner of the object that was added to the realm. To find the available users, query the DBA_USERS view, described in *Oracle Database Reference*.
object_name	Object name. (The wildcard % is allowed. See "Object Name" under "Creating Realm-Secured Objects" on page 4-4 for exceptions to the wildcard %.) You can also use the DVSYS.DBMS_MACUTL G_ALL_OBJECT constant. To find objects that are secured by existing realms, query the DVSYS.DBA_DV_REALM_OBJECT view, described in "DBA_DV_REALM_OBJECT View" on page 10-22.
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 %.) You can also use the DVSYS.DBMS_MACUTL G_ALL_OBJECT constant.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_OBJECT_FROM_REALM(
  realm_name   => 'Performance Statistics Realm',
  object_owner => 'SYS',
  object_name  => 'GATHER_SYSTEM_STATISTICS',
  object_type  => 'ROLE');
END;
/
```

DELETE_REALM Procedure

The DELETE_REALM procedure deletes a realm but does not remove its associated objects and authorizations. Before you delete a realm, you can locate its associated objects by querying the DVSYS.DBA_DV_REALM_OBJECT view, described in"Oracle Database Vault Data Dictionary Views" on page 10-11.

If you want to remove the associated objects and authorizations as well as the realm, see "DELETE_REALM_CASCADE Procedure" on page 11-8.

Syntax

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

Parameters

Table 11–7 DELETE_REALM Parameter

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21

Example

```
EXEC DVSYS.DBMS_MACADM.DELETE_REALM('Performance Statistics Realm');
```

DELETE_REALM_CASCADE Procedure

The DELETE_REALM_CASCADE procedure deletes a realm, including its related Database Vault configuration information that specifies who is authorized (DVSYS.DBA_DV_REALM_AUTH view) and what objects are protected (DVSYS.DBA_DV_REALM_OBJECT view). It does not delete the actual database objects or users. To find a listing of the realm-related objects, query the DVSYS.DBA_DV_REALM view. To find its authorizations, query DVSYS.DBA_DV_REALM_AUTH. Both are described under "Oracle Database Vault Data Dictionary Views" on page 10-11.

Syntax

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

Parameters

Table 11–8 DELETE_REALM_CASCADE Parameter

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21

Example

```
EXEC DVSYS.DBMS_MACADM.DELETE_REALM_CASCADE('Performance Statistics Realm');
```

RENAME_REALM Procedure

The RENAME_REALM procedure renames a realm. The name change takes effect everywhere the realm is used.

Syntax

```
DVSYS.DBMS_MACADM.RENAME_REALM(
  realm_name IN VARCHAR2,
  new_name   IN VARCHAR2);
```

Parameters

Table 11–9 RENAME_REALM Parameters

Parameter	Description
realm_name	Current realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21
new_name	New realm name, up to 90 characters in mixed-case.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.RENAME_REALM(
  realm_name => 'Performance Statistics Realm',
  new_name   => 'Sector 2 Performance Statistics Realm');
END;
/
```

UPDATE_REALM Procedure

The UPDATE_REALM procedure updates a realm.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_REALM(
  realm_name    IN VARCHAR2,
  description   IN VARCHAR2,
  enabled       IN VARCHAR2,
  audit_options IN NUMBER DEFAULT NULL);
```

Parameters

Table 11–10 UPDATE_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21
description	Description of the purpose of the realm, up to 1024 characters in mixed-case.

Table 11–10 (Cont.) UPDATE_REALM Parameters

Parameter	Description
enabled	Y (yes) enables realm checking; N (no) disables it. The default is Y. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_YES ■ G_NO
audit_options	Optional. The default is previous audit option setting, which you can check by querying the DVSYS.DBA_DV_REALM data dictionary view. Specify one of the following options 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 You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_REALM_AUDIT_OFF ■ G_REALM_AUDIT_SUCCESS ■ G_REALM_AUDIT_FAIL ■ G_REALM_AUDIT_FAIL + G_REALM_AUDIT_SUCCESS

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_REALM(
  realm_name    => 'Sector 2 Performance Statistics Realm',
  description   => 'Realm to measure performance for Sector 2 applications',
  enabled       => 'Y',
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_FAIL + G_REALM_AUDIT_SUCCESS);
END;
/
```

UPDATE_REALM_AUTH Procedure

The UPDATE_REALM_AUTH procedure updates the authorization of a user or role to access a realm.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_REALM_AUTH(
  realm_name    IN VARCHAR2,
  grantee       IN VARCHAR2,
  rule_set_name IN VARCHAR2,
  auth_options  IN NUMBER);
```

Parameters

Table 11–11 UPDATE_REALM_AUTH Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DVSYS.DBA_DV_REALM view, described in "DBA_DV_REALM View" on page 10-21
grantee	User or role name. To find the available users and roles, query the DBA_USERS and DBA_ROLES views, described in *Oracle Database Reference*. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view, described in DBA_DV_REALM_AUTH View on page 10-22. To find existing secure application roles used in privilege management, query the DVSYS.DBA_DV_ROLE view, described in "DBA_DV_ROLE View" on page 10-23.
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, query the DVSYS.DBA_DV_RULE_SET view. To find rules that are associated with the rule sets, query the DBA_DB_RULE_SET_RULE view. Both are described in "Oracle Database Vault Data Dictionary Views" on page 10-11.
auth_options	Optional. Specify one of the following options to authorize the realm: ■ 0: 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. ■ 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 multiple owners. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_REALM_AUTH_PARTICIPANT (default) ■ G_REALM_AUTH_OWNER

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_REALM_AUTH(
  realm_name    => 'Sector 2 Performance Statistics Realm',
  grantee       => 'SYSADM',
  rule_set_name => 'Check Conf Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

Rule Set Procedures Within DVSYS.DBMS_MACADM

Table 11–12 lists procedures within the `DVSYS.DBMS_MACADM` package that you can use to configure rule sets.

See Also:

- Chapter 5, "Configuring Rule Sets" for detailed information about rule sets
- Chapter 13, "Using the DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility procedures that you can use with the rule set procedures
- Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces," for PL/SQL interfaces that you can use with rules

Table 11–12 DVSYS.DBMS_MACADM Rule Set Configuration Procedures

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

ADD_RULE_TO_RULE_SET Procedure

The `ADD_RULE_TO_RULE_SET` procedure adds a rule to a rule set, and lets you specify whether to have the rule be checked when the rule set is evaluated.

Syntax

```
DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name  IN VARCHAR2,
  rule_name      IN VARCHAR2,
  rule_order     IN NUMBER,
  enabled        IN VARCHAR2);
```

Parameters

Table 11–13 ADD_RULE_TO_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.
rule_name	Rule to add to the rule set. To find existing rules, query the DVSYS.DBA_DV_RULE view, described in "DBA_DV_RULE View" on page 10-23. To find rules that have been associated with rule sets, use DVSYS.DBA_DV_RULE_SET_RULE, described in "DBA_DV_RULE View" on page 10-23.
rule_order	Does not apply to this release, but you must include a value for the ADD_RULE_TO_RULE_SET procedure to work. Enter 1.
enabled	Optional. Determines whether the rule should be checked when the rule set is evaluated. Possible values are: ■ Y (yes; default) ■ N (no) You can also enter the following DVSYS.DBMS_MACUTL constants: ■ G_YES (default) ■ G_NO See Table 13–1 on page 13-1 for more information.

Examples

The following example adds a rule to a rule set, and by omitting the enabled parameter, automatically enables the rule to be checked when the rule set is evaluated.

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Restrict DROP TABLE operations'),
  rule_order    => 1);
END;
/
```

This example adds the rule to the rule set but disables rule checking.

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Check UPDATE operations',
  rule_order    => 1,
  enabled       => DBMS_MACUTL.G_NO);
END;
/
```

CREATE_RULE Procedure

The CREATE_RULE procedure creates a rule. After you create a rule, you can add it to a rule set.

Syntax

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

Parameters

Table 11–14 CREATE_RULE Parameters

Parameter	Description
rule_name	Rule name, up to 90 characters in mixed-case. Spaces are allowed. To find existing rules in the current database instance, query the DVSYS.DBA_DV_RULE view, described in "DBA_DV_RULE View" on page 10-23. To find rules that have been associated with rule sets, query DVSYS.DBA_DV_RULE_SET_RULE, described in "DBA_DV_RULE_SET_RULE View" on page 10-25.
rule_expr	PL/SQL BOOLEAN expression. If the expression contains quotation marks, do not use double quotation marks. Instead, use two single quotation marks. Enclose the entire expression within single quotation marks. For example: `'TO_CHAR(SYSDATE,''HH24'') = ''12'''` See "Creating a New Rule" on page 5-5 for more information on rule expressions.

Example

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

CREATE_RULE_SET Procedure

The CREATE_RULE_SET procedure creates a rule set. After you create a rule set, you can use the CREATE_RULE and ADD_RULE_TO_RULE_SET procedures to create and add rules to the rule set.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    IN VARCHAR2,
  description      IN VARCHAR2,
  enabled          IN VARCHAR2,
  eval_options     IN NUMBER,
  audit_options    IN NUMBER,
  fail_options     IN NUMBER,
  fail_message     IN VARCHAR2,
  fail_code        IN NUMBER,
  handler_options  IN NUMBER,
```

```
  handler          IN VARCHAR2);
```

Parameters

Table 11–15 CREATE_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name, up to 90 characters in mixed-case. Spaces are allowed. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.
description	Description of the purpose of the rule set, up to 1024 characters in mixed-case.
enabled	Y (yes) enables the rule set; N (no) disables it. The default is Y. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_YES ■ G_NO
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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_RULESET_EVAL_ALL ■ G_RULESET_EVAL_ANY
audit_options	Select one of the following settings: ■ 0: Disables auditing for the rule set ■ 1: Creates an audit record when a rule set violation occurs ■ 2: Creates an audit record for a successful rule set evaluation ■ 3: Creates an audit record for both successful and failed rule set evaluations You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_RULESET_AUDIT_OFF ■ G_RULESET_AUDIT_FAIL ■ G_RULESET_AUDIT_SUCCESS ■ G_RULESET_AUDIT_FAIL + G_RULESET_AUDIT_SUCCESS 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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_RULESET_FAIL_SHOW ■ G_RULESET_FAIL_SILENT 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.

Table 11–15 (Cont.) CREATE_RULE_SET Parameters

Parameter	Description
fail_code	Enter a negative number in the range of -20000 to -20999, to associate with the fail_message parameter.
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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_RULESET_HANDLER_OFF ■ G_RULESET_HANDLER_FAIL ■ G_RULESET_HANDLER_SUCCESS See "Error Handling Options" on page 5-4 for more information.
handler	Name of the PL/SQL function or procedure that defines the custom event handler logic. See "Error Handling Options" on page 5-4 for more information.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => 'Y',
  eval_options     => 2,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL + DBMS_MACUTL.G_RULESET_
AUDIT_SUCCESS,
  fail_options     => 2,
  fail_message     => '',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler          => 'dbavowner.email_alert');
END;
/
```

See Also: Example 13–2, "Creating a Rule Set Using DVSYS.DBMS_MACUTL Constants" on page 13-5

DELETE_RULE Procedure

The DELETE_RULE procedure deletes a rule.

Syntax

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

Parameter

Table 11–16 DELETE_RULE Parameter

Parameter	Description
rule_name	Rule name. To find existing rules in the current database instance, query the DVSYS.DBA_DV_RULE view, described in "DBA_DV_RULE View" on page 10-23. To find rules that have been associated with rule sets, query DVSYS.DBA_DV_RULE_SET_RULE, described in "DBA_DV_RULE_SET_RULE View" on page 10-25.

Example

```
EXEC DVSYS.DBMS_MACADM.DELETE_RULE('Check UPDATE operations');
```

DELETE_RULE_FROM_RULE_SET Procedure

The DELETE_RULE_FROM_RULE_SET procedure deletes a rule from a rule set.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
  rule_set_name IN VARCHAR2,
  rule_name     IN VARCHAR2);
```

Parameters

Table 11–17 DELETE_RULE_FROM_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.
rule_name	Rule to remove from the rule set. To find existing rules in the current database instance, query the DVSYS.DBA_DV_RULE view, described in "DBA_DV_RULE View" on page 10-23. To find rules that have been associated with rule sets, query DVSYS.DBA_DV_RULE_SET_RULE, described in "DBA_DV_RULE_SET_RULE View" on page 10-25.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Check UPDATE operations');
END;
/
```

DELETE_RULE_SET Procedure

The DELETE_RULE_SET procedure deletes a rule set.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_RULE_SET(
  rule_set_name IN VARCHAR2);
```

Parameters

Table 11–18 DELETE_RULE_SET Parameter

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.

Example

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

RENAME_RULE Procedure

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

Syntax

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

Parameters

Table 11–19 RENAME_RULE Parameters

Parameter	Description
rule_name	Current rule name. To find existing rules in the current database instance, query the DVSYS.DBA_DV_RULE view, described in "DBA_DV_RULE View" on page 10-23. To find rules that have been associated with rule sets, query DVSYS.DBA_DV_RULE_SET_RULE, described in "DBA_DV_RULE_SET_RULE View" on page 10-25.
new_name	New rule name, up to 90 characters in mixed-case.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.RENAME_RULE(
  rule_name  => 'Check UPDATE operations',
  new_name   => 'Check Sector 2 Processes');
END;
/
```

RENAME_RULE_SET Procedure

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

Syntax

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

Parameters

Table 11–20 RENAME_RULE_SET Parameters

Parameter	Description
rule_set_name	Current rule set name. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.
new_name	New rule set name, up to 90 characters in mixed-case. Spaces are allowed.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.RENAME_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  new_name      => 'Limit Sector 2 Access');
END;
/
```

SYNC_RULES Procedure

The SYNC_RULES procedure synchronizes the rules in Oracle Database Vault and Advanced Queuing Rules engine. You must perform this operation immediately after you roll back an ADD_RULE_TO_RULESET, a DELETE_RULE, or an UPDATE_RULE operation.

Syntax

```
DVSYS.DBMS_MACADM.SYNC_RULES();
```

Parameters

None.

Example

```
EXEC DVSYS.DBMS_MACADM.SYNC_RULES();
```

UPDATE_RULE Procedure

The UPDATE_RULE procedure updates a rule.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_RULE(
  rule_name  IN VARCHAR2,
  rule_expr  IN VARCHAR2);
```

Parameters

Table 11–21 UPDATE_RULE Parameters

Parameter	Description
rule_name	Rule name. To find existing rules in the current database instance, query the DVSYS.DBA_DV_RULE view, described in "DBA_DV_RULE View" on page 10-23. To find rules that have been associated with rule sets, query DVSYS.DBA_DV_RULE_SET_RULE, described in "DBA_DV_RULE_SET_RULE View" on page 10-25.
rule_expr	PL/SQL BOOLEAN expression. If the expression contains quotation marks, do not use double quotation marks. Instead, use two single quotation marks. Enclose the entire expression within single quotation marks. For example: `'TO_CHAR(SYSDATE,''HH24'') = ''12'''` See "Creating a New Rule" on page 5-5 for more information on rule expressions. To find existing rule expressions, query the DVSYS.DBA_DV_RULE view.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_RULE(
  rule_name  => 'Check UPDATE operations',
  rule_expr  =>'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SYSADM'' AND
               (
                 UPPER(SYS_CONTEXT(''USERENV'',''MODULE'')) LIKE ''APPSRVR%'' OR
                 UPPER(SYS_CONTEXT(''USERENV'',''MODULE'')) LIKE ''DBAPP%'' )'
               );
END;
/
```

UPDATE_RULE_SET Procedure

The UPDATE_RULE_SET procedure updates a rule set.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_RULE_SET(
  rule_set_name    IN VARCHAR2,
  description      IN VARCHAR2,
  enabled          IN VARCHAR2,
  eval_options     IN NUMBER,
  audit_options    IN NUMBER,
  fail_options     IN NUMBER,
  fail_message     IN VARCHAR2,
  fail_code        IN NUMBER,
  handler_options  IN NUMBER,
  handler          IN VARCHAR2);
```

Parameters

Table 11–22 UPDATE_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.
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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_YES ■ G_NO
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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_RULESET_EVAL_ALL ■ G_RULESET_EVAL_ANY
audit_options	Select one of the following settings: ■ 0: Disables auditing for the rule set ■ 1: Creates an audit record when a rule set violation occurs ■ 2: Creates an audit record for a successful rule set evaluation ■ 3: Creates an audit record for both successful and failed rule set evaluations You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_RULESET_AUDIT_OFF ■ G_RULESET_AUDIT_FAIL ■ G_RULESET_AUDIT_SUCCESS ■ G_RULESET_AUDIT_FAIL + G_RULESET_AUDIT_SUCCESS 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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_RULESET_FAIL_SHOW ■ G_RULESET_FAIL_SILENT 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 parameter.

Table 11–22 (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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_RULESET_HANDLER_OFF ■ G_RULESET_HANDLER_FAIL ■ G_RULESET_HANDLER_SUCCESS See "Error Handling Options" on page 5-4 for more information.
handler	Name of the PL/SQL function or procedure that defines the custom event handler logic. See "Error Handling Options" on page 5-4 for more information.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => 'Y',
  eval_options     => 2,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'Access denied!',
  fail_code        => -20900,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
  handler          => '');
END;
/
```

Command Rule Procedures Within DVSYS.DBMS_MACADM

Table 11–23 lists procedures 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 Chapter 13, "Using the DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility procedures that you can use with the command rule procedures.

Table 11–23 DVSYS.DBMS_MACADM Command Rule Configuration Procedures

Procedure	Description
CREATE_COMMAND_RULE Procedure	Creates a command rule, associates it with a rule set, and lets you enable the command rule for rule checking with a rule set
DELETE_COMMAND_RULE Procedure	Drops a command rule declaration
UPDATE_COMMAND_RULE Procedure	Updates a command rule declaration

CREATE_COMMAND_RULE Procedure

The CREATE_COMMAND_RULE procedure creates a command rule, associates it with a rule set, and lets you enable the command rule for rule checking with a rule set.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_COMMAND_RULE(
  command         IN VARCHAR2,
  rule_set_name   IN VARCHAR2,
  object_owner    IN VARCHAR2,
  object_name     IN VARCHAR2,
  enabled         IN VARCHAR2);
```

Parameters

Table 11–24 CREATE_COMMAND_RULE Parameters

Parameter	Description
command	SQL statement to protect. See also the following: ■ "DBA_DV_COMMAND_RULE View" on page 10-14 for a listing of existing command rules ■ "SQL Statements That Can Be Protected by Command Rules" on page 6-3 for a listing of available SQL statements that you can use ■ *Oracle Database SQL Reference* for more information about SQL statements
rule_set_name	Name of rule set to associate with this command rule. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.
object_owner	Database schema owner for this command rule. The wildcard % is allowed. To find the available users, query 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, query the ALL_OBJECTS view, described in *Oracle Database Reference*.
enabled	Y (yes) enables the command rule; N (no) disables it. The default is Y. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_YES ■ G_NO

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'SELECT',
  rule_set_name   => 'Limit Sector 2 Access',
  object_owner    => 'SYSADM',
  object_name     => 'EMP_DATA',
```

```
  enabled         => 'DBMS_MACUTL.G_YES');
END;
/
```

DELETE_COMMAND_RULE Procedure

The DELETE_COMMAND_RULE procedure drops a command rule declaration.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_COMMAND_RULE(
  command      IN VARCHAR2,
  object_owner IN VARCHAR2,
  object_name  IN VARCHAR2);
```

Parameters

Table 11–25 DELETE_COMMAND_RULE Parameters

Parameter	Description
command	SQL statement the command rule protects. To find available command rules, query the DVSYS.DBA_DV_COMMAND_RULE view, described in "DBA_DV_COMMAND_RULE View" on page 10-14
object_owner	Database schema owner for this command rule. To find the available users in the current database instance, query the DBA_USERS view, described in *Oracle Database Reference*. See also "Object Owner" in "Creating 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, query the ALL_OBJECTS view, described in *Oracle Database Reference*.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_COMMAND_RULE(
  command      => 'SELECT',
  object_owner => 'SYSADM',
  object_name  => 'EMP_DATA');
END;
/
```

UPDATE_COMMAND_RULE Procedure

The UPDATE_COMMAND_RULE procedure updates a command rule declaration.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE(
  command        IN VARCHAR2,
  rule_set_name  IN VARCHAR2,
  object_owner   IN VARCHAR2,
  object_name    IN VARCHAR2,
  enabled        IN VARCHAR2);
```

Parameters

Table 11–26 UPDATE_COMMAND_RULE Parameters

Parameter	Description
command	SQL statement to protect. See also the following: ■ "DBA_DV_COMMAND_RULE View" on page 10-14 for a listing of existing command rules ■ "SQL Statements That Can Be Protected by Command Rules" on page 6-3 for a listing of available SQL statements that you can use ■ *Oracle Database SQL Reference* for more information about SQL statements
rule_set_name	Name of rule set to associate with this command rule. To find existing rule sets in the current database instance, query the DVSYS.DBA_DV_RULE_SET view, described in "Oracle Database Vault Data Dictionary Views" on page 10-11.
object_owner	Database schema owner for this command rule. To find the available users, query 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, query the ALL_OBJECTS view, described in *Oracle Database Reference*.
enabled	Y (yes) enables the command rule; N (no) disables it. The default is Y. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_YES ■ G_NO

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'SELECT',
  rule_set_name   => 'Limit Sector 2 Access',
  object_owner    => 'SYSADM',
  object_name     => '%',
  enabled         => 'DBMS_MACUTL.G_NO');
END;
/
```

Factor Procedures and Functions Within DVSYS.DBMS_MACADM

Table 11–27 lists procedures and functions within the `DVSYS.DBMS_MACADM` package that you can use to configure factors.

See Also:

- Chapter 7, "Configuring Factors" for detailed information about factors
- Chapter 13, "Using the DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility procedures that you can use with the factor procedures
- Chapter 14, "Using the Oracle Database Vault PL/SQL Interfaces," for a set of PL/SQL interfaces that you can use with factors

Table 11–27 DVSYS.DBMS_MACADM Factor Configuration Procedures

Procedure	Description
ADD_FACTOR_LINK Procedure	Specifies a parent-child relationship for two factors.
ADD_POLICY_FACTOR Procedure	Specifies that the label for a factor contributes to the Oracle Label Security label for a policy
CHANGE_IDENTITY_FACTOR Procedure	Associates an identity with a different factor
CHANGE_IDENTITY_VALUE Procedure	Updates the value of an identity
CREATE_DOMAIN_IDENTITY Procedure	Adds an Oracle Real Application Clusters (Oracle RAC) database node to the domain factor identities and labels it according to the Oracle Label Security policy
CREATE_FACTOR Procedure	Creates a factor
CREATE_FACTOR_TYPE Procedure	Creates a factor type
CREATE_IDENTITY Procedure	Creates an identity
CREATE_IDENTITY_MAP Procedure	Defines a set of tests that are used to derive the identity of a factor from the value of linked child factors (subfactors)
DELETE_FACTOR Procedure	Deletes a factor
DELETE_FACTOR_LINK Procedure	Removes a parent-child relationship for two factors
DELETE_FACTOR_TYPE Procedure	Deletes a factor type
DELETE_IDENTITY Procedure	Removes an identity
DELETE_IDENTITY_MAP Procedure	Removes an identity map from a factor
DROP_DOMAIN_IDENTITY Procedure	Removes an Oracle RAC database node from a domain
GET_INSTANCE_INFO Function	Returns information from the `SYS.V_$INSTANCE` system table about the current database instance; returns a `VARCHAR2` value
GET_SESSION_INFO Function	Returns information from the `SYS.V_$SESSION` system table for the current session; returns a `VARCHAR2` value
RENAME_FACTOR Procedure	Renames a factor. The name change takes effect everywhere the factor is used.
RENAME_FACTOR_TYPE Procedure	Renames a factor type. The name change takes effect everywhere the factor type is used.

Table 11–27 (Cont.) DVSYS.DBMS_MACADM Factor Configuration Procedures

Procedure	Description
UPDATE_FACTOR Procedure	Updates a factor
UPDATE_FACTOR_TYPE Procedure	Updates the description of a factor type
UPDATE_IDENTITY Procedure	Updates the trust level of a factor identity

ADD_FACTOR_LINK Procedure

The ADD_FACTOR_LINK procedure specifies a parent-child relationship for two factors.

Syntax

```
DVSYS.DBMS_MACADM.ADD_FACTOR_LINK(
  parent_factor_name IN VARCHAR2,
  child_factor_name  IN VARCHAR2,
  label_indicator    IN VARCHAR2);
```

Parameters

Table 11–28 ADD_FACTOR_LINK Parameters

Parameter	Description
parent_factor_name	Parent factor name. To find existing parent and child factors in the current database instance, query the DVSYS.DBA_DV_FACTOR_LINK view, described in "DBA_DV_FACTOR_LINK View" on page 10-16.
child_factor_name	Child factor name.
label_indicator	Indicates that the child factor being linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Specify either Y (for Yes) or N (for No). You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_YES ■ G_NO To find the Oracle Label Security policies and labels associated with factors, query the following views, described in "Oracle Database Vault Data Dictionary Views" on page 10-11: ■ DVSYS.DBA_DV_MAC_POLICY: Lists Oracle Label Security policies defined in the current database instance. ■ DVSYS.DBA_DV_MAC_POLICY_FACTOR: Lists the factors that are associated with Oracle Label Security policies for the current database instance. ■ DVSYS.DBA_DV_POLICY_LABEL: Lists the Oracle Label Security label for each factor identifier in the DVSYS.DBA_DV_IDENTITY view for each policy.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_FACTOR_LINK(
  parent_factor_name => 'HQ_ClientID',
  child_factor_name  => 'Div1_ClientID',
  label_indicator    => 'Y');
END;
/
```

ADD_POLICY_FACTOR Procedure

The ADD_POLICY_FACTOR procedure specifies that the label for a factor contributes to the Oracle Label Security label for a policy.

Syntax

```
DVSYS.DBMS_MACADM.ADD_POLICY_FACTOR(
  policy_name  IN VARCHAR2,
  factor_name  IN VARCHAR2);
```

Parameters

Table 11–29 ADD_POLICY_FACTOR Parameters

Parameter	Description
policy_name	Oracle Label Security policy name. To find the policies defined in the current database instance, query the DVSYS.DBA_DV_MAC_POLICY view, described in "DBA_DV_MAC_POLICY View" on page 10-19. To find factors that are associated with Oracle Label Security policies, query DVSYS.DBA_DV_MAC_POLICY_FACTOR, described in "DBA_DV_MAC_POLICY_FACTOR View" on page 10-19.
factor_name	Factor name. To find existing factors, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_POLICY_FACTOR(
  policy_name  => 'AccessData',
  factor_name  => 'Sector2_ClientID');
END;
/
```

CHANGE_IDENTITY_FACTOR Procedure

The CHANGE_IDENTITY_FACTOR procedure associates an identity with a different factor.

Syntax

```
DVSYS.DBMS_MACADM.CHANGE_IDENTITY_FACTOR(
  factor_name     IN VARCHAR2,
  value           IN VARCHAR2,
  new_factor_name IN VARCHAR2);
```

Parameters

Table 11–30 CHANGE_IDENTITY_FACTOR Parameters

Parameter	Description
factor_name	Current factor name. To find existing factors, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15

Table 11–30 (Cont.) CHANGE_IDENTITY_FACTOR Parameters

Parameter	Description
value	Value of the identity to update. To find existing identities for each factor in the current database instance, query the DVSYS.DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17. To find current identity mappings, query the DVSYS.DBA_DV_IDENTITY_MAP view, described in "DBA_DV_IDENTITY_MAP View" on page 10-18.
new_factor_name	Name of the factor to associate with the identity.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CHANGE_IDENTITY_FACTOR(
  factor_name     => 'Sector2_ClientID',
  value           => 'intranet',
  new_factor_name => 'Sector4_ClientID');
END;
/
```

CHANGE_IDENTITY_VALUE Procedure

The CHANGE_IDENTITY_FACTOR procedure updates the value of an identity.

Syntax

```
DVSYS.DBMS_MACADM.CHANGE_IDENTITY_VALUE(
  factor_name  IN VARCHAR2,
  value        IN VARCHAR2,
  new_value    IN VARCHAR2);
```

Parameters

Table 11–31 CHANGE_IDENTITY_VALUE Parameters

Parameter	Description
factor_name	Factor name. To find existing factors, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15
value	Current value associated with the identity. To find existing identities for each factor in the current database instance, query the DVSYS.DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17. To find current identity mappings, query the DVSYS.DBA_DV_IDENTITY_MAP view, described in "DBA_DV_IDENTITY_MAP View" on page 10-18.
new_value	New identity value, up to 1024 characters in mixed-case.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CHANGE_IDENTITY_VALUE(
  factor_name  => 'Sector2_ClientID',
  value        => 'remote',
  new_value    => 'intranet');
```

```
END;
/
```

CREATE_DOMAIN_IDENTITY Procedure

The CREATE_DOMAIN_IDENTITY procedure adds an Oracle RAC database node to the domain factor identities and labels it according to the Oracle Label Security policy.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_DOMAIN_IDENTITY(
  domain_name  IN VARCHAR2,
  domain_host  IN VARCHAR2,
  policy_name  IN VARCHAR2 DEFAULT NULL,
  domain_label IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 11–32 CREATE_DOMAIN_IDENTITY Parameters

Parameter	Description
domain_name	Name of the domain to which to add the host. To find the logical location of the database within the network structure within a distributed database system, run the DVF.F$DATABASE_DOMAIN function, described in "Oracle Database Vault PL/SQL Factor Functions" on page 14-5.
domain_host	Oracle Real Application Clusters host name being added to the domain. To find host name of a database, run the DVF.F$DATABASE_HOSTNAME function, described in "Oracle Database Vault PL/SQL Factor Functions" on page 14-5.
policy_name	Oracle Label Security policy name. If you omit the policy name, then the domain is not associated with any policy. To find the available policies, query the DVSYS.DBA_DV_MAC_POLICY view, described in "DBA_DV_MAC_POLICY View" on page 10-19.
domain_label	Name of the domain to which to add the Oracle Label Security policy.

Examples

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_DOMAIN_IDENTITY(
  domain_name  => 'example',
  domain_host  => 'mydom_host',
  policy_name  => 'AccessData',
  domain_label => 'sensitive');
END;
/
```

CREATE_FACTOR Procedure

The CREATE_FACTOR procedure creates a factor. After you create a factor, you can give it an identity by using the CREATE_IDENTITY procedure, described in "CREATE_IDENTITY Procedure" on page 11-34.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_FACTOR(
  factor_name       IN VARCHAR2,
  factor_type_name  IN VARCHAR2,
```

```
  description       IN VARCHAR2,
  rule_set_name     IN VARCHAR2,
  get_expr          IN VARCHAR2,
  validate_expr     IN VARCHAR2,
  identify_by       IN NUMBER,
  labeled_by        IN NUMBER,
  eval_options      IN NUMBER,
  audit_options     IN NUMBER,
  fail_options      IN NUMBER);
```

Parameters

Table 11–33 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, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.
factor_type_name	Type of the factor, up to 30 characters in mixed-case, without spaces. To find existing factor types, query the DBA_DV_FACTOR_TYPE view, described in "DBA_DV_FACTOR_TYPE View" on page 10-17.
description	Description of the purpose of the factor, up to 1024 characters in mixed-case.
rule_set_name	Rule set name if you want to use a rule set to control when and how a factor identity is set. To find existing rule sets, query the DVSYS.DBA_DV_RULE_SET view, described in "Oracle Database Vault Data Dictionary Views" on page 10-11. 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 procedure to validate the factor. This is a valid PL/SQL expression that returns a Boolean value (TRUE or FALSE) to validate the identity of the factor. See "Validation Method" on page 7-8 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 You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_IDENTIFY_BY_CONSTANT ■ G_IDENTIFY_BY_METHOD ■ G_IDENTIFY_BY_FACTOR ■ G_IDENTIFY_BY_CONTEXT See "Factor Identification" on page 7-5 for more information.

Table 11–33 (Cont.) CREATE_FACTOR Parameters

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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_LABELED_BY_SELF ■ G_LABELED_BY_FACTORS See "Factor Labeling" on page 7-7 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 You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_EVAL_ON_SESSION ■ G_EVAL_ON_ACCESS See "Evaluation" on page 7-7 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 procedure returns an error. ■ POWER(2,4): Audits if the validation procedure is false. ■ POWER(2,5): Audits if there is no trust level set. ■ POWER(2,6): Audits if the trust level is negative. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_AUDIT_OFF ■ G_AUDIT_ALWAYS ■ G_AUDIT_ON_GET_ERROR ■ G_AUDIT_ON_GET_NULL ■ G_AUDIT_ON_VALIDATE_ERROR ■ G_AUDIT_ON_VALIDATE_FALSE ■ G_AUDIT_ON_TRUST_LEVEL_NULL ■ G_AUDIT_ON_TRUST_LEVEL_NEG See "Audit Options" on page 7-9 for more information.

Table 11–33 (Cont.) CREATE_FACTOR Parameters

Parameter	Description
fail_options	Options for reporting factor errors: ■ POWER(2,0): Shows an error message. ■ POWER(2,1): Does not show an error message. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_FAIL_WITH_MESSAGE ■ G_FAIL_SILENTLY See "Error Options" on page 7-10 for more information.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'Limit_DBA_Access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => 2,
  labeled_by        => 0,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_SESSION,
  audit_options     => DBMS_MACUTL.G_AUDIT_OFF,
  fail_options      => DBMS_MACUTL.G_FAIL_SILENTLY);
END;
/
```

CREATE_FACTOR_TYPE Procedure

The CREATE_FACTOR_TYPE procedure creates a user-defined factor type.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_FACTOR_TYPE(
  name        IN VARCHAR2,
  description IN VARCHAR2);
```

Parameters

Table 11–34 CREATE_FACTOR_TYPE Parameters

Parameter	Description
name	Factor type name, up to 30 characters in mixed-case, without spaces. To find existing factor types, query the DVSYS.DBA_DV_FACTOR_TYPE view, described in "DBA_DV_FACTOR_TYPE View" on page 10-17.
description	Description of the purpose of the factor type, up to 1024 characters in mixed-case.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_FACTOR_TYPE(
  name        => 'Sector2Instance',
  description => 'Checks DB instances used in Sector 2');
END;
```

```
/
```

CREATE_IDENTITY Procedure

The CREATE_IDENTITY procedure assigns an identity and an associated trust level for a given factor. After you create a factor, you must assign it an identity.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_IDENTITY(
  factor_name  IN VARCHAR2,
  value        IN VARCHAR2,
  trust_level  IN NUMBER);
```

Parameters

Table 11–35 CREATE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.
value	The actual value of the factor, up to 1024 characters in mixed-case. For example, the identity of an IP_Address factor could be the IP address of 192.0.2.12.
trust_level	Number that indicates the magnitude of trust relative to other identities for the same factor. In general, the higher the trust level number is set, the greater the trust. A trust level of 10 indicates "very trusted." Negative trust levels are not trusted. See "Creating and Configuring a Factor Identity" on page 7-11 for more information about trust levels and label security.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_IDENTITY(
  factor_name  => 'Sector2_ClientID',
  value        => 'intranet',
  trust_level  => 5);
END;
/
```

CREATE_IDENTITY_MAP Procedure

The CREATE_IDENTITY_MAP procedure defines a set of tests that are used to derive the identity of a factor from the value of linked child factors (subfactors).

Syntax

```
DVSYS.DBMS_MACADM.CREATE_IDENTITY_MAP(
  identity_factor_name  IN VARCHAR2,
  identity_factor_value IN VARCHAR2,
  parent_factor_name    IN VARCHAR2,
  child_factor_name     IN VARCHAR2,
  operation             IN VARCHAR2,
  operand1              IN VARCHAR2,
  operand2              IN VARCHAR2);
```

Parameters

Table 11–36 CREATE_IDENTITY_MAP Parameters

Parameter	Description
identity_factor_name	Factor the identity map is for. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "Oracle Database Vault Data Dictionary Views" on page 10-11.
identity_factor_value	Value the factor assumes if the identity map evaluates to TRUE. To find existing factor identities, query the DVSYS.DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17. To find current factor identity mappings, use DVSYS.DBA_DV_IDENTITY_MAP, described in "DBA_DV_IDENTITY_MAP View" on page 10-18.
parent_factor_name	The parent factor link to which the map is related. To find existing parent-child factor mappings, query the DVSYS.DBA_DV_IDENTITY_MAP view, described in "DBA_DV_IDENTITY_MAP View" on page 10-18.
child_factor_name	The child factor link to which the map is related.
operation	Relational operator for the identity map (for example, <, >, =, and so on).
operand1	Left operand for the relational operator; refers to the low value you enter.
operand2	Right operand for the relational operator; refers to the high value you enter.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_IDENTITY_MAP(
  identity_factor_name  => 'Sector2_ClientID',
  identity_factor_value => 'intranet',
  parent_factor_name    => 'HQ_ClientID',
  child_factor_name     => 'Div1_ClientID',
  operation             => '<',
  operand1              => '192.0.2.50',
  operand2              => '192.0.2.100');
END;
/
```

DELETE_FACTOR Procedure

The DELETE_FACTOR procedure deletes a factor.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_FACTOR(
  factor_name IN VARCHAR2);
```

Parameters

Table 11–37 DELETE_FACTOR Parameter

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.

Example

```
EXEC DVSYS.DBMS_MACADM.DELETE_FACTOR('Sector2_ClientID');
```

DELETE_FACTOR_LINK Procedure

The DELETE_FACTOR_LINK procedure removes a parent-child relationship for two factors.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_FACTOR_LINK(
  parent_factor_name IN VARCHAR2,
  child_factor_name  IN VARCHAR2);
```

Parameters

Table 11–38 DELETE_FACTOR_LINK Parameters

Parameter	Description
parent_factor_name	Factor name. To find factors that are used in parent-child mappings in the current database instance, query the DVSYS.DBA_DV_FACTOR_LINK view, described in "DBA_DV_FACTOR_LINK View" on page 10-16.
child_factor_name	Factor name.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_FACTOR_LINK(
  parent_factor_name => 'HQ_ClientID',
  child_factor_name  => 'Div1_ClientID');
END;
/
```

DELETE_FACTOR_TYPE Procedure

The DELETE_FACTOR_TYPE procedure deletes a factor type.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_FACTOR_TYPE(
  name IN VARCHAR2);
```

Parameters

Table 11–39 DELETE_FACTOR_TYPE Parameters

Parameter	Description
name	Factor type name. To find existing factor types, query the DVSYS.DBA_DV_FACTOR_TYPE view, described in "DBA_DV_FACTOR_TYPE View" on page 10-17.

Example

```
EXEC DVSYS.DBMS_MACADM.DELETE_FACTOR_TYPE('Sector2Instance');
```

DELETE_IDENTITY Procedure

The DELETE_IDENTITY procedure removes an identity from an existing factor.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_IDENTITY(
  factor_name IN VARCHAR2,
  value       IN VARCHAR2);
```

Parameters

Table 11–40 DELETE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.
value	Identity value associated with the factor. To find the identities for each factor in the current database instance, query the DVSYS.DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_IDENTITY(
  factor_name => 'Sector2_ClientID',
  value       => 'intranet');
END;
/
```

DELETE_IDENTITY_MAP Procedure

The DELETE_IDENTITY_MAP procedure removes an identity map for a factor.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_IDENTITY_MAP(
  identity_factor_name  IN VARCHAR2,
  identity_factor_value IN VARCHAR2,
  parent_factor_name    IN VARCHAR2,
  child_factor_name     IN VARCHAR2,
  operation             IN VARCHAR2,
  operand1              IN VARCHAR2,
```

```
  operand2              IN VARCHAR2);
```

Parameters

Table 11–41 DELETE_IDENTITY_MAP Parameters

Parameter	Description
identity_factor_name	Factor the identity map is for. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.
identity_factor_value	Value the factor assumes if the identity map evaluates to TRUE. To find existing factor identities, query the DVSYS.DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17. To find current factor identity mappings, query DVSYS.DBA_DV_IDENTITY_MAP, described in "DBA_DV_IDENTITY_MAP View" on page 10-18.
parent_factor_name	The parent factor link to which the map is related. To find existing parent-child factors, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR_LINK View" on page 10-16.
child_factor_name	The child factor to which the map is related.
operation	Relational operator for the identity map (for example, <, >, =, and so on).
operand1	Left (low value) operand for the relational operator.
operand2	Right (high value) operand for the relational operator.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_IDENTITY_MAP(
  identity_factor_name  => 'Sector2_ClientID',
  identity_factor_value => 'intranet',
  parent_factor_name    => 'HQ_ClientID',
  child_factor_name     => 'Div1_ClientID',
  operation             => '<',
  operand1              => '192.0.2.10',
  operand2              => '192.0.2.15');
END;
/
```

DROP_DOMAIN_IDENTITY Procedure

The DROP_DOMAIN_IDENTITY procedure removes an Oracle Real Application Clusters database node from a domain.

Syntax

```
DVSYS.DBMS_MACADM.DROP_DOMAIN_IDENTITY(
  domain_name  IN VARCHAR2,
  domain_host  IN VARCHAR2);
```

Parameters

Table 11–42 DROP_DOMAIN_IDENTITY Parameters

Parameter	Description
domain_name	Name of the domain to which the host was added. To find the domain of a database as specified by the DB_DOMAIN initialization parameter, run the DVF.F$DATABASE_DOMAIN function, described in "Oracle Database Vault PL/SQL Factor Functions" on page 14-5.
domain_host	Oracle Real Application Clusters host name being that was added to the domain. To find the host name for a specified database, run the DVF.F$DATABASE_HOSTNAME function, described in "Oracle Database Vault PL/SQL Factor Functions" on page 14-5.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DROP_DOMAIN_IDENTITY(
  domain_name  => 'example',
  domain_host  => 'mydom_host');
END;
/
```

GET_INSTANCE_INFO Function

The GET_INSTANCE_INFO function returns information from the SYS.V_$INSTANCE system table about the current database instance. The V$INSTANCE data dictionary view displays information from this table. See *Oracle Database Reference* for more information.

Syntax

```
DVSYS.DBMS_MACADM.GET_INSTANCE_INFO(
  p_parameter IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 11–43 GET_INSTANCE_INFO Parameter

Parameter	Description
p_parameter	Column name in the SYS.V_$INSTANCE system table.

Example

```
DECLARE
 instance_var varchar2 := null;
BEGIN
 instance_var = DVSYS.DBMS_MACADM.GET_INSTANCE_INFO('INSTANCE_NAME');
END;
/
```

GET_SESSION_INFO Function

The GET_SESSION_INFO function returns information from the SYS.V_$SESSION system table for the current session. The V$SESSION data dictionary view displays information from this table. See *Oracle Database Reference* for more information.

Syntax

```
DVSYS.DBMS_MACADM.GET_SESSION_INFO(
  p_parameter IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 11–44 GET_SESSION_INFO Parameter

Parameter	Description
p_parameter	Column name in the SYS.V_$SESSION system table.

Example

```
DECLARE
 session_var varchar2 := null;
BEGIN
 session_var = DVSYS.DBMS_MACADM.GET_SESSION_INFO('PROCESS');
END;
/
```

RENAME_FACTOR Procedure

The RENAME_FACTOR procedure renames a factor. The name change takes effect everywhere the factor is used.

Syntax

```
DVSYS.DBMS_MACADM.RENAME_FACTOR(
  factor_name     IN VARCHAR2,
  new_factor_name IN VARCHAR2);
```

Parameters

Table 11–45 RENAME_FACTOR Parameters

Parameter	Description
factor_name	Current factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.
new_factor_name	New factor name, up to 30 characters in mixed-case, without spaces.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.RENAME_FACTOR(
  factor_name     => 'Sector2_ClientID',
  new_factor_name => 'Sector2_Clients');
END;
/
```

RENAME_FACTOR_TYPE Procedure

The RENAME_FACTOR procedure renames a factor type. The name change takes effect everywhere the factor type is used.

Syntax

```
DVSYS.DBMS_MACADM.RENAME_FACTOR_TYPE(
  old_name  IN VARCHAR2,
  new_name  IN VARCHAR2);
```

Parameters

Table 11–46 RENAME_FACTOR_TYPE Parameters

Parameter	Description
old_name	Current factor type name. To find existing factor types in the current database instance, query the DVSYS.DBA_DV_FACTOR_TYPE view, described in "DBA_DV_FACTOR_TYPE View" on page 10-17.
new_name	New factor type name, up to 30 characters in mixed-case, without spaces.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.RENAME_FACTOR_TYPE(
  old_name  => 'Sector2Instance',
  new_name  => 'Sector2DBInstance');
END;
/
```

UPDATE_FACTOR Procedure

The UPDATE_FACTOR procedure updates the description of a factor type.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_FACTOR(
  factor_name       IN VARCHAR2,
  factor_type_name  IN VARCHAR2,
  description       IN VARCHAR2,
  rule_set_name     IN VARCHAR2,
  get_expr          IN VARCHAR2,
  validate_expr     IN VARCHAR2,
  identify_by       IN NUMBER,
  labeled_by        IN NUMBER,
  eval_options      IN NUMBER,
  audit_options     IN NUMBER,
  fail_options      IN NUMBER);
```

Parameters

Table 11–47 UPDATE_FACTOR

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.

Table 11–47 (Cont.) UPDATE_FACTOR

Parameter	Description
factor_type_name	Factor type name. To find existing factor types, query the DVSYS.DBA_DV_FACTOR_TYPE view, described in "DBA_DV_FACTOR_TYPE View" on page 10-17.
description	Description of the purpose of the factor, up to 1024 characters in mixed-case.
rule_set_name	Name of the rule set used to control when and how a factor identity is set. To find existing rule sets, query the DVSYS.DBA_DV_RULE_SET view, described in "Oracle Database Vault Data Dictionary Views" on page 10-11. 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 procedure to validate factor. This is a valid PL/SQL expression that returns a Boolean value (TRUE or FALSE) to validate the identity of the factor. See "Validation Method" on page 7-8 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 You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_IDENTIFY_BY_CONSTANT ■ G_IDENTIFY_BY_METHOD ■ G_IDENTIFY_BY_FACTOR ■ G_IDENTIFY_BY_CONTEXT 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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_LABELED_BY_SELF ■ G_LABELED_BY_FACTORS See "Factor Labeling" on page 7-7 for more information.

Table 11–47 (Cont.) UPDATE_FACTOR

Parameter	Description
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 You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_EVAL_ON_SESSION ■ G_EVAL_ON_ACCESS See "Evaluation" on page 7-7 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 procedure returns an error. ■ POWER(2,4): Audits if the validation procedure is false. ■ POWER(2,5): Audits if there is no trust level set. ■ POWER(2,6): Audits if the trust level is negative. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_AUDIT_OFF ■ G_AUDIT_ALWAYS ■ G_AUDIT_ON_GET_ERROR ■ G_AUDIT_ON_GET_NULL ■ G_AUDIT_ON_VALIDATE_ERROR ■ G_AUDIT_ON_VALIDATE_FALSE ■ G_AUDIT_ON_TRUST_LEVEL_NULL ■ G_AUDIT_ON_TRUST_LEVEL_NEG See "Audit Options" on page 7-9 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. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_FAIL_WITH_MESSAGE ■ G_FAIL_SILENTLY See "Error Options" on page 7-10 for more information.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'Limit_DBA_Access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
```

```
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => 2,
  labeled_by        => 0,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_ACCESS,
  audit_options     => DBMS_MACUTL.G_AUDIT_ALWAYS,
  fail_options      => DBMS_MACUTL.G_FAIL_WITH_MESSAGE);
END;
/
```

UPDATE_FACTOR_TYPE Procedure

The UPDATE_FACTOR_TYPE procedure updates a factor type.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_FACTOR_TYPE(
  name         IN VARCHAR2,
  description  IN VARCHAR2);
```

Parameters

Table 11–48 UPDATE_FACTOR_TYPE Parameters

Parameter	Description
name	Factor type name. To find existing factor types in the current database instance, query the DVSYS.DBA_DV_FACTOR_TYPE view, described in "DBA_DV_FACTOR_TYPE View" on page 10-17.
description	Description of the purpose of the factor type, up to 1024 characters in mixed case.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_FACTOR_TYPE(
  name        => 'Sector2DBInstance',
  description => 'Checks DB instances used in Sector 2');
END;
/
```

UPDATE_IDENTITY Procedure

The UPDATE_IDENTITY procedure updates the trust level of a factor identity.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_IDENTITY(
  factor_name  IN VARCHAR2,
  value        IN VARCHAR2,
  trust_level  IN NUMBER);
```

Parameters

Table 11–49 UPDATE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DBSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15. To find factors that have identities, query DVSYS.DBA_DV_IDENTITY, described in "DBA_DV_IDENTITY View" on page 10-17.
value	New factor identity, up to 1024 characters in mixed-case. For example, the identity of an IP_Address factor could be the IP address of 192.0.2.12.
trust_level	Number that indicates the magnitude of trust relative to other identities for the same factor. In general, the higher the trust level number is set, the greater the trust. A trust level of 10 indicates "very trusted." Negative trust levels are not trusted. See "Creating and Configuring a Factor Identity" on page 7-11 for more information about trust levels and label security.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_IDENTITY(
  factor_name  => 'Sector2_ClientID',
  value        => 'intranet',
  trust_level  => 10);
END;
/
```

Secure Application Role Procedures Within DVSYS.DBMS_MACADM

Table 11–50 lists procedures 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 Chapter 13, "Using the DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility procedures that you can use with the secure application role procedures.

Table 11–50 DVSYS.DBMS_MACADM Secure Application Role Configuration Procedures

Procedure	Description
CREATE_ROLE Procedure	Creates an Oracle Database Vault secure application role
DELETE_ROLE Procedure	Deletes an Oracle Database Vault secure application role
RENAME_ROLE Procedure	Renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used
UPDATE_ROLE Procedure	Updates a Oracle Database Vault secure application role

CREATE_ROLE Procedure

The CREATE_ROLE procedure creates an Oracle Database Vault secure application role.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_ROLE(
  role_name      IN VARCHAR2,
  enabled        IN VARCHAR2,
  rule_set_name  IN VARCHAR2);
```

Parameters

Table 11–51 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. To find existing secure application roles in the current database instance, query the DVSYS.DBA_DV_ROLE view, described in "DBA_DV_ROLE View" on page 10-23.
enabled	Y (yes) makes the role available for enabling; N (no) prevents the role from being enabled. The default is Y You can also use the following DVSYS.DBMS_MACUTL constants: ▪ G_YES ▪ G_NO
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, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_ROLE(
  role_name      => 'Sector2_APP_MGR',
  enabled        => 'Y',
  rule_set_name  => 'Check App2 Access');
END;
/
```

DELETE_ROLE Procedure

The DELETE_ROLE procedure deletes an Oracle Database Vault secure application role.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_ROLE(
  role_name IN VARCHAR2);
```

Parameters

Table 11–52 DELETE_ROLE Parameter

Parameter	Description
role_name	Role name. To find existing secure application roles in the current database instance, query the DVSYS.DBA_DV_ROLE view, described in "DBA_DV_ROLE View" on page 10-23.

Example

```
EXEC DVSYS.DBMS_MACADM.DELETE_ROLE('SECT2_APP_MGR');
```

RENAME_ROLE Procedure

The RENAME_ROLE procedure renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.

Syntax

```
DVSYS.DBMS_MACADM.RENAME_ROLE(
  role_name      IN VARCHAR2,
  new_role_name  IN VARCHAR2);
```

Parameters

Table 11–53 RENAME_ROLE Parameters

Parameter	Description
role_name	Current role name. Preferably, enter the role name in upper case letters, though you are not required to do so. To find existing secure application roles in the current database instance, query the DVSYS.DBA_DV_ROLE view, described in "DBA_DV_ROLE View" on page 10-23.
new_role_name	Role name, up to 30 characters, in uppercase, with no spaces.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.RENAME_ROLE(
  role_name      => 'SECT2_APP_MGR',
  new_role_name  => 'SECT2_SYSADMIN', );
END;
/
```

UPDATE_ROLE Procedure

The UPDATE_ROLE procedure updates a Oracle Database Vault secure application role.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_ROLE(
  role_name      IN VARCHAR2,
  enabled        IN VARCHAR2,
  rule_set_name  IN VARCHAR2);
```

Parameters

Table 11–54 UPDATE_ROLE Parameters

Parameter	Description
role_name	Role name. To find existing secure application roles in the current database instance, query the DVSYS.DBA_DV_ROLE view, described in "DBA_DV_ROLE View" on page 10-23.
enabled	Y (yes) makes the role available for enabling; N (no) prevents the role from being enabled. The default is Y. You can also use the following DVSYS.DBMS_MACUTL constants: ■ G_YES ■ G_NO
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, query the DVSYS.DBA_DV_RULE_SET view, described in "DBA_DV_RULE_SET View" on page 10-24.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_ROLE(
  role_name      => 'SECT2_SYSADMIN',
  enabled        => 'Y',
  rule_set_name  => 'System Access Controls');
END;
/
```

Oracle Label Security Policy Procedures Within DVSYS.DBMS_MACADM

Table 11–55 lists procedures 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 Chapter 13, "Using the DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility procedures that you can use with the Oracle Label Security policy procedures.

Table 11–55 DVSYS.DBMS_MACADM Oracle Label Security Configuration Procedures

Procedure	Description
CREATE_MAC_POLICY Procedure	Specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label
CREATE_POLICY_LABEL Procedure	Labels an identity within an Oracle Label Security policy
DELETE_MAC_POLICY_CASCADE Procedure	Deletes all Oracle Database Vault objects related to an Oracle Label Security policy

Table 11–55 (Cont.) DVSYS.DBMS_MACADM Oracle Label Security Configuration Procedures

Procedure	Description
DELETE_POLICY_FACTOR Procedure	Removes the factor from contributing to the Oracle Label Security label
DELETE_POLICY_LABEL Procedure	Removes the label from an identity within an Oracle Label Security policy
UPDATE_MAC_POLICY Procedure	Specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label

CREATE_MAC_POLICY Procedure

The `CREATE_MAC_POLICY` procedure specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_MAC_POLICY(
  policy_name  IN VARCHAR2,
  algorithm    IN VARCHAR2);
```

Parameters

Table 11–56 CREATE_MAC_POLICY Parameters

Parameter	Description
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DVSYS.DBA_DV_MAC_POLICY` view, described in "DBA_DV_MAC_POLICY View" on page 10-19.
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. Enter the code listed in Table 11–57 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 11–57 Oracle Label Security Merge Algorithm Codes

Code	Value
`HUU`	Maximum Level/Union/Union
`HIU`	Maximum Level/Intersection/Union
`HMU`	Maximum Level/Minus/Union
`HNU`	Maximum Level/Null/Union
`HUI`	Maximum Level/Union/Intersection
`HII`	Maximum Level/Intersection/Intersection
`HMI`	Maximum Level/Minus/Intersection
`HNI`	Maximum Level/Null/Intersection
`HUM`	Maximum Level/Union/Minus
`HIM`	Maximum Level/Intersection/Minus

Table 11–57 (Cont.) Oracle Label Security Merge Algorithm Codes

Code	Value
HMM	Maximum Level/Minus/Minus
HNM	Maximum Level/Null/Minus
HUN	Maximum Level/Union/Null
HIN	Maximum Level/Intersection/Null
HMN	Maximum Level/Minus/Null
HNN	Maximum Level/Null/Null
LUU	Minimum Level/Union/Union
LIU	Minimum Level/Intersection/Union
LMU	Minimum Level/Minus/Union
LNU	Minimum Level/Null/Union
LUI	Minimum Level/Union/Intersection
LII	Minimum Level/Intersection/Intersection
LMI	Minimum Level/Minus/Intersection
LNI	Minimum Level/Null/Intersection
LUM	Minimum Level/Union/Minus
LIM	Minimum Level/Intersection/Minus
LMM	Minimum Level/Minus/Minus
LNM	Minimum Level/Null/Minus
LUN	Minimum Level/Union/Null
LIN	Minimum Level/Intersection/Null
LMN	Minimum Level/Minus/Null
LNN	Minimum Level/Null/Null

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_MAC_POLICY(
  policy_name  => 'Access Locations',
  algorithm    => 'HUU');
END;
/
```

CREATE_POLICY_LABEL Procedure

The CREATE_POLICY_LABEL procedure labels an identity within an Oracle Label Security policy.

Syntax

```
DVSYS.DBMS_MACADM.CREATE_POLICY_LABEL(
  identity_factor_name   IN VARCHAR2,
  identity_factor_value  IN VARCHAR2,
  policy_name            IN VARCHAR2,
  label                  IN VARCHAR2);
```

Parameters

Table 11–58 CREATE_POLICY_LABEL Parameters

Parameter	Description
identity_factor_name	Name of factor being labeled. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15. To find factors that are associated with Oracle Label Security policies, use DVSYS.DBA_DV_MAC_POLICY_FACTOR, described in "DBA_DV_MAC_POLICY_FACTOR View" on page 10-19. See also "Label Security Policy Factors" on page 9-9 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, query the DVSYS.DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17.
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DVSYS.DBA_DV_MAC_POLICY view, described in "DBA_DV_MAC_POLICY View" on page 10-19.
label	Oracle Label Security label name. To find existing policy labels for factor identifiers, query the DVSYS.DBA_DV_POLICY_LABEL view, described in "DBA_DV_POLICY_LABEL View" on page 10-20.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_POLICY_LABEL(
  identity_factor_name   => 'App_Host_Name',
  identity_factor_value  => 'Sect2_Fin_Apps',
  policy_name            => 'Access Locations',
  label                  => 'Sensitive');
END;
/
```

DELETE_MAC_POLICY_CASCADE Procedure

The DELETE_MAC_POLICY_CASCADE procedure deletes all Oracle Database Vault objects related to an Oracle Label Security policy.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_MAC_POLICY_CASCADE(
  policy_name IN VARCHAR2);
```

Parameters

Table 11–59 DELETE_MAC_POLICY_CASCADE Parameter

Parameter	Description
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DVSYS.DBA_DV_MAC_POLICY view, described in "DBA_DV_MAC_POLICY View" on page 10-19.

Example

```
EXEC DVSYS.DBMS_MACADM.DELETE_MAC_POLICY_CASCADE('Access Locations');
```

DELETE_POLICY_FACTOR Procedure

The DELETE_POLICY_FACTOR procedure removes the factor from contributing to the Oracle Label Security label.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_POLICY_FACTOR(
  policy_name  IN VARCHAR2,
  factor_name  IN VARCHAR2);
```

Parameters

Table 11–60 DELETE_POLICY_FACTOR Parameters

Parameter	Description
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DVSYS.DBA_DV_MAC_POLICY view, described in "DBA_DV_MAC_POLICY View" on page 10-19.
factor_name	Name of factor associated with the Oracle Label Security label. To find factors that are associated with Oracle Label Security policies, query DVSYS.DBA_DV_MAC_POLICY_FACTOR, described in "DBA_DV_MAC_POLICY_FACTOR View" on page 10-19.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_POLICY_FACTOR(
  policy_name  => 'Access Locations',
  factor_name  => 'App_Host_Name');
END;
/
```

DELETE_POLICY_LABEL Procedure

The DELETE_POLICY_LABEL procedure removes the label from an identity within an Oracle Label Security policy.

Syntax

```
DVSYS.DBMS_MACADM.DELETE_POLICY_LABEL(
  identity_factor_name   IN VARCHAR2,
  identity_factor_value  IN VARCHAR2,
  policy_name            IN VARCHAR2,
```

```
  label                 IN VARCHAR2);
```

Parameters

Table 11–61 DELETE_POLICY_LABEL Parameters

Parameter	Description
identity_factor_name	Name of the factor that was labeled. To find existing factors in the current database instance that are associated with Oracle Label Security policies, query DVSYS.DBA_DV_MAC_POLICY_FACTOR, described in "DBA_DV_MAC_POLICY_FACTOR View" on page 10-19. See also "Label Security Policy Factors" on page 9-9 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, query the DVSYS.DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17.
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DVSYS.DBA_DV_MAC_POLICY view, described in "DBA_DV_MAC_POLICY View" on page 10-19.
label	Oracle Label Security label name. To find existing policy labels for factor identifiers, query the DVSYS.DBA_DV_POLICY_LABEL view, described in "DBA_DV_POLICY_LABEL View" on page 10-20.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.DELETE_POLICY_LABEL(
  identity_factor_name   => 'App_Host_Name',
  identity_factor_value  => 'Sect2_Fin_Apps',
  policy_name            => 'Access Locations',
  label                  => 'Sensitive');
END;
/
```

UPDATE_MAC_POLICY Procedure

The UPDATE_MAC_POLICY procedure specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Syntax

```
DVSYS.DBMS_MACADM.UPDATE_MAC_POLICY(
  policy_name  IN VARCHAR2,
  algorithm    IN VARCHAR2);
```

Parameters

Table 11–62 UPDATE_MAC_POLICY

Parameter	Description
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DVSYS.DBA_DV_MAC_POLICY` view, described in "DBA_DV_MAC_POLICY View" on page 10-19.
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. See Table 11–57 on page 11-49 for listing of the available algorithms. For more information on label-merging algorithms, see *Oracle Label Security Administrator's Guide*.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_MAC_POLICY(
  policy_name  => 'Access Locations',
  algorithm    => 'LUI');
END;
/
```

12

Using the DVSYS.DBMS_MACSEC_ROLES Package

This chapter contains:

- About the DVSYS.DBMS_MACSEC_ROLES Package
- CAN_SET_ROLE Function
- SET_ROLE Procedure

About the DVSYS.DBMS_MACSEC_ROLES Package

You can modify your applications to use the procedures 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 Chapter 13, "Using the DVSYS.DBMS_MACUTL Package" for a set of general-purpose utility procedures that you can use with the secure application role procedures.

Table 12–1 lists the `DVSYS.DBMS_MACSEC_ROLES` package function and procedure.

Table 12–1 DVSYS.DBMS_MACSEC_ROLES Oracle Label Security Configuration Procedures

Function or Procedure	Description
CAN_SET_ROLE Function	Checks whether the user invoking the method is authorized to use the specified Oracle Database Vault secure application role. Returns a `BOOLEAN` value.
SET_ROLE Procedure	Issues the `SET ROLE` statement for an Oracle Database Vault secure application role.

CAN_SET_ROLE Function

The `CAN_SET_ROLE` function checks whether the user invoking the method is authorized to use the specified Oracle Database Vault secure application role.

Syntax

```
DVSYS.DBMS_MACSEC_ROLES.CAN_SET_ROLE(
  p_role IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 12–2 CAN_SET_ROLE Parameter

Parameter	Description
p_role	Role name. To find existing secure application roles in the current database instance, query the DVSYS.DBA_DV_ROLE view, described in "DBA_DV_ROLE View" on page 10-23.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DVSYS.DBMS_MACSEC_ROLES.SET_ROLE('SECTOR2_APP_MGR')
  THEN DBMS_OUTPUT.PUT_LINE('SECTOR2_APP_MGR' is enabled.')
 END IF;
END;
/
```

SET_ROLE Procedure

The SET_ROLE procedure 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

```
DVSYS.DBMS_MACSEC_ROLES.SET_ROLE(
  p_role IN VARCHAR2);
```

Parameters

Table 12–3 SET_ROLE Parameter

Parameter	Description
p_role	Role name. To find existing secure application roles in the current database instance, query the DVSYS.DBA_DV_ROLE view, described in "DBA_DV_ROLE View" on page 10-23.

Example

```
EXEC DVSYS.DBMS_MACSEC_ROLES.SET_ROLE('SECTOR2_APP_MGR');
```

You can enter the name of the role in any case (for example, Sector2_APP_MGR).

13

Using the DVSYS.DBMS_MACUTL Package

This chapter contains:

- About the DVSYS.DBMS_MACUTL Package
- DVSYS.DBMS_MACUTL Constants
- Procedures and Functions Within the DVSYS.DBMS_MACUTL Package

About the DVSYS.DBMS_MACUTL Package

The `DVSYS.DBMS_MACUTL` package provides a set of general purpose utility procedures and functions that you can use throughout the application code you write for Oracle Database Vault. This package is available to all users.

DVSYS.DBMS_MACUTL Constants

This section contains:

- DVSYS.DBMS_MACUTL Listing of Constants
- Examples of Using the DVSYS.DBMS_MACUTL Constants

DVSYS.DBMS_MACUTL Listing of Constants

Table 13–1 summarizes constant (that is, fields) descriptions for the `DVSYS.DBMS_MACUTL` package. You can use these constants with any of the Oracle Database Vault PL/SQL packages. Many of these constants have equivalents in the Oracle Database Vault package. For example, the `enabled` parameter, which is available in several procedures, can accept either `Y` (for Yes) or the constant `G_YES`. Choosing one over the other is a matter of personal preference. They both have the same result.

Table 13–1 DVSYS.DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
`G_ALL_OBJECT`	`VARCHAR2(1)`	Used with the realm API `object_name` and `object_type` parameters as a wildcard to indicate all object names or all object types.
`G_AUDIT_ALWAYS`	`NUMBER`	Used with the factor API `audit_options` parameter to enable an audit.
`G_AUDIT_OFF`	`NUMBER`	Used with the factor API `audit_options` parameter to disable auditing.

Table 13–1 (Cont.) DVSYS.DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_AUDIT_ON_GET_ERROR	NUMBER	Used with the factor API audit_options parameter to audit if the expression specified in the get_expr parameter returns an error.
G_AUDIT_ON_GET_NULL	NUMBER	Used with the factor API audit_options parameter to audit if the expression in the get_expr field is null.
G_AUDIT_ON_TRUST_LEVEL_NEG	NUMBER	Used with the factor API audit_options parameter to audit if the trust level is negative.
G_AUDIT_ON_TRUST_LEVEL_NULL	NUMBER	Used with the factor API audit_options parameter to audit if no trust level exists.
G_AUDIT_ON_VALIDATE_ERROR	NUMBER	Used with the factor API audit_options parameter to audit if the validation function returns an error.
G_AUDIT_ON_VALIDATE_FALSE	NUMBER	Used with the factor API audit_options parameter to audit if validation function is false.
G_EVAL_ON_ACCESS	NUMBER	Used with the factor API eval_options parameter to reevaluate the factor each time it is accessed.
G_EVAL_ON_SESSION	NUMBER	Used with the factor API eval_options parameter to evaluate the factor only once, when the user logs in to the session.
G_FAIL_SILENTLY	NUMBER	Used with the fail_options parameter to fail and show no error message.
G_FAIL_WITH_MESSAGE	NUMBER	Used with the fail_options parameter to fail and show an error message.
G_IDENTIFY_BY_CONSTANT	NUMBER	Used with the factor API identify_by parameter: Fixed value in PL/SQL expression defined in the get_expr parameter.
G_IDENTIFY_BY_CONTEXT	NUMBER	Used with the factor API identify_by parameter to indicate context.
G_IDENTIFY_BY_FACTOR	NUMBER	Used with the factor API identify_by parameter for subfactors through the factor_link$ table.
G_IDENTIFY_BY_METHOD	NUMBER	Used with the factor API identify_by parameter: Expression in get_expr field
G_IDENTIFY_BY_RULESET	NUMBER	Used with the factor API identify_by parameter: Expression and Rule Set with the factor_expr$ table

Table 13–1 (Cont.) DVSYS.DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_LABELED_BY_FACTORS	NUMBER	Used with the factor API labeled_by parameter to derive the label from subfactor and merge algorithm.
G_LABELED_BY_SELF	NUMBER	Used with the factor API labeled_by parameter to label the factor identities.
G_MAX_SESSION_LABEL	VARCHAR2(30)	This is the highest label a user could set based on the factors. It does not consider the label for a user.
G_MIN_POLICY_LABEL	VARCHAR2(30)	The label to which a factor with a null label defaults.
G_NO	VARCHAR2(1)	Used with the following APIs: ■ The factor API label_indicator parameter to indicate that a child factor linked to a parent factor does not contribute to the label of the parent factor in an Oracle Label Security integration. ■ Any API that uses the enabled parameter.
G_OLS_SESSION_LABEL	VARCHAR2(30)	The Oracle Label Security session label for a user at the time init_session is run.
G_REALM_AUDIT_FAIL	NUMBER	Used with the realm API audit_options parameter to audit when the realm is violated.
G_REALM_AUDIT_OFF	NUMBER	Used with the realm API audit_options parameter to disable auditing.
G_REALM_AUDIT_SUCCESS	NUMBER	Used with the realm API audit_options parameter: Audit on successful realm access
G_REALM_AUTH_OWNER	NUMBER	Used with the realm API auth_options parameter to set the realm authorization to Owner.
G_REALM_AUTH_PARTICIPANT	NUMBER	Used with the realm API auth_options parameter to set the realm authorization to Participant.
G_RULESET_AUDIT_FAIL	NUMBER	Used with the rule set API audit_options parameter to audit on rule set failure.
G_RULESET_AUDIT_OFF	NUMBER	Used with the rule set API audit_options parameter to disable auditing.
G_RULESET_AUDIT_SUCCESS	NUMBER	Used with the rule set API audit_options parameter to audit on rule set success.
G_RULESET_EVAL_ALL	NUMBER	Used with the rule set API eval_options parameter to enable the rule set to succeed if all rules evaluate to true.

Table 13–1 (Cont.) DVSYS.DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_RULESET_EVAL_ANY	NUMBER	Used with the rule set API eval_options parameter to succeed if any of the rules evaluate to true.
G_RULESET_FAIL_SHOW	NUMBER	Used with the rule set API fail_options parameter to show an error message if the rule set fails.
G_RULESET_FAIL_SILENT	NUMBER	Used with the rule set API fail_options parameter to not show an error message if the rule set fails.
G_RULESET_HANDLER_FAIL	NUMBER	Used with the rule set API handler_options parameter to call a handler (specified by the handler parameter) if the rule set fails.
G_RULESET_HANDLER_OFF	NUMBER	Used with the rule set API handler_options parameter to disable calls to a handler or if no handler is used.
G_RULESET_HANDLER_SUCCESS	NUMBER	Used with the rule set API handler_options parameter to call a handler if the rule set succeeds.
G_USER_POLICY_LABEL	VARCHAR2(30)	This is what Oracle Label Security has decided the user's label should be set to after factoring in the preceding values.
G_YES	VARCHAR2(1)	Used with the following APIs: ■ The factor API label_indicator parameter to indicate that a child factor linked to a parent factor contributes to the label of the parent factor in an Oracle Label Security integration. ■ Any API that uses the enabled parameter.

Examples of Using the DVSYS.DBMS_MACUTL Constants

Example 13–1 shows how to use the G_YES and G_REALM_AUDIT_FAIL DBMS_MACUTL constants when creating a realm.

Example 13–1 Creating a Realm Using DVSYS.DBMS_MACUTL Constants

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Performance Statistics Realm',
  description   => 'Realm to measure performance',
  enabled       =>  DVSYS.DBMS_MACUTL.G_YES,
  audit_options =>  DVSYS.DBMS_MACUTL.G_REALM_AUDIT_FAIL);
END;
/
```

Example 13–2 shows how to use several DVSYS.DBMS_MACUTL constants when creating a rule set.

Example 13–2 Creating a Rule Set Using DVSYS.DBMS_MACUTL Constants

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name     => 'Limit_DBA_Access',
  description       => 'DBA access through predefined processes',
  enabled           => 'Y',
  eval_options      => DBMS_MACUTL.G_RULESET_EVAL_ALL,
  audit_options     => DBMS_MACUTL.G_RULESET_AUDIT_FAIL,
  fail_options      => DVSYS.DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message      => 'Rule Set Limit_DBA_Access has failed.',
  fail_code         => -20000,
  handler_options   => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler           => 'dbavowner.email_alert');
END;
/
```

Example 13–3 shows how to use constants when creating a factor.

Example 13–3 Creating a Factor Using DVSYS.DBMS_MACUTL Constants

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'DB_access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_FACTOR,
  labeled_by        => DVSYS.DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DVSYS.DBMS_MACUTL.G_EVAL_ON_SESSION,
  audit_options     => DBMS_MACUTL.G_AUDIT_ALWAYS,
  fail_options      => DVSYS.DBMS_MACUTL.G_FAIL_SILENTLY);
END;
/
```

Procedures and Functions Within the DVSYS.DBMS_MACUTL Package

Table 13–2 lists the procedures and functions in the DVSYS.DBMS_MACUTL package. You can use these procedures or functions as standalone code, or within rule expressions. The examples in this section show a mixture of using both.

Table 13–2 DVSYS.DBMS_MACUTL Utility Functions

Procedure or Function	Descriptions
CHECK_DVSYS_DML_ALLOWED Procedure	Verifies that public-packages are not being bypassed by users updating the Oracle Database Vault configuration
GET_CODE_VALUE Function	Looks up the value for a code within a code group
GET_SECOND Function	Returns the seconds in Oracle SS format (00-59). Useful for rule expressions based on time data
GET_MINUTE Function	Returns the minute in Oracle MI format (00–59). Useful for rule expressions based on time data

Table 13–2 (Cont.) DVSYS.DBMS_MACUTL Utility Functions

Procedure or Function	Descriptions
GET_HOUR Function	Returns the month in Oracle HH24 format (00–23). Useful for rule expressions based on time data
GET_DAY Function	Returns the day in Oracle DD format (01–31). Useful for rule expressions based on time data
GET_MONTH Function	Returns the month in Oracle MM format (01–12). Useful for rule expressions based on time data
GET_YEAR Function	Returns the year in Oracle YYYY format (0001–9999). Useful for rule expressions based on time data
IS_ALPHA Function	Checks whether the character is alphabetic
IS_DIGIT Function	Checks whether the character is numeric
IS_DVSYS_OWNER Function	Determines whether a user is authorized to manage the Oracle Database Vault configuration
IS_OLS_INSTALLED Function	Returns an indicator regarding whether Oracle Label Security is installed
IS_OLS_INSTALLED_VARCHAR Function	Returns an indicator regarding whether Oracle Label Security is installed
USER_HAS_OBJECT_PRIVILEGE Function	Checks whether a user or role may access an object through an object privilege grant
USER_HAS_ROLE Function	Checks whether a user has a role privilege, directly or indirectly (through another role)
USER_HAS_ROLE_VARCHAR Function	Checks whether a user has a role privilege, directly or indirectly (through another role)
USER_HAS_SYSTEM_PRIVILEGE Function	Checks whether a user has a system privilege, directly or indirectly (through a role)

CHECK_DVSYS_DML_ALLOWED Procedure

The `CHECK_DVSYS_DML_ALLOWED` procedure verifies that public packages are not being bypassed by users updating the Oracle Database Vault configuration.

Syntax

```
DVSYS.DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED(
  p_user IN VARCHAR2 DEFAULT USER);
```

Parameter

Table 13–3 CHECK_DVSYS_DML_ALLOWED Parameter

Parameter	Description
`p_user`	User performing the operation. To find existing users in the current database instance, query the following views: ▪ `DBA_USERS`: Finds available users for the current database instance. See *Oracle Database Reference*. ▪ `DVSYS.DBA_DV_REALM_AUTH`: Finds the authorization of a particular user or role. See "DBA_DV_REALM_AUTH View" on page 10-22. ▪ `DVSYS.DBA_DV_ROLE`: Finds existing secure application roles used in privilege management. See "DBA_DV_ROLE View" on page 10-23.

Example

User SYSTEM fails the check:

```
EXEC DVSYS.DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED('system');

ERROR at line 1:
ORA-47920: Authorization failed for user system to perform this operation
ORA-06512: at "DVSYS.DBMS_MACUTL", line 23
ORA-06512: at "DVSYS.DBMS_MACUTL", line 372
ORA-06512: at "DVSYS.DBMS_MACUTL", line 508
ORA-06512: at "DVSYS.DBMS_MACUTL", line 572
ORA-06512: at line 1
```

User lbrown_dvowner, who has the DV_OWNER role, passes the check:

```
EXEC DVSYS.DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED('lbrown_dvowner');

PL/SQL procedure successfully completed.
```

GET_CODE_VALUE Function

The GET_CODE_VALUE function finds the value for a code within a code group, and then returns a VARCHAR2 value.

Syntax

```
DVSYS.DBMS_MACUTL.GET_CODE_VALUE(
  p_code_group IN VARCHAR2,
  p_code       IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 13–4 GET_CODE_VALUE Parameters

Parameter	Description
p_code_group	Code group (for example, AUDIT_EVENTS or BOOLEAN). To find available code groups in the current database instance, query the DVSYS.DBA_DV_CODE view, described in "DBA_DV_CODE View" on page 10-12.
p_code	ID of the code. This ID is listed when you run the DVSYS.DBA_DV_CODE view.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Label Algorithm for Maximum Level/Union/Null',
  rule_expr => 'DVSYS.DBMS_MACUTL.GET_CODE_VALUE(''LABEL_ALG'', ''HUN'') =
''Union''');
END;
/
```

GET_SECOND Function

The GET_SECOND function returns the seconds in Oracle SS (seconds) format (00–59), and then returns a NUMBER value. It is useful for rule expressions based on time data.

Syntax

```
DVSYS.DBMS_MACUTL.GET_SECOND(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 13–5 GET_SECOND Parameter

Parameter	Description
p_date	Date in SS format (for example, 59). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  seconds number;
BEGIN
  seconds := DVSYS.DBMS_MACUTL.GET_SECOND(TO_DATE('03-APR-2009 6:56 PM',
  'dd-mon-yyyy hh:mi PM'));
  DBMS_OUTPUT.PUT_LINE('Seconds: '||seconds);
END;
/
```

This example, which uses a fixed date and time, returns the following:

```
Seconds: 56
```

GET_MINUTE Function

The GET_MINUTE function returns the minute in Oracle MI (minute) format (00–59), in a NUMBER value. It is useful for rule expressions based on time data.

Syntax

```
DVSYS.DBMS_MACUTL.GET_MINUTE(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 13–6 GET_MINUTE Parameter

Parameter	Description
p_date	Date in MI format (for example, 30, as in 2:30). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  minute number;
BEGIN
  minute := DVSYS.DBMS_MACUTL.GET_MINUTE(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Minute: '||minute);
END;
/
```

Output similar to the following appears:

```
Minute: 17
```

GET_HOUR Function

The GET_HOUR function returns the hour in Oracle HH24 (hour) format (00–23), in a NUMBER value. It is useful for rule expressions based on time data.

Syntax

```
DVSYS.DBMS_MACUTL.GET_HOUR(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 13–7 GET_HOUR Parameter

Parameter	Description
p_date	Date in HH24 format (for example, 14 for 2:00 p.m.). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  hours number;
BEGIN
  hours := DVSYS.DBMS_MACUTL.GET_HOUR(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Hour: '||hours);
END;
/
```

Output similar to the following appears:

```
Hour: 12
```

GET_DAY Function

The GET_DAY function returns the day in Oracle DD (day) format (01–31), in a NUMBER value. It is useful for rule expressions based on time data.

Syntax

```
DVSYS.DBMS_MACUTL.GET_DAY(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 13–8 GET_DAY Parameter

Parameter	Description
p_date	Date in DD format (for example, 01 for the first day of the month). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  day number;
BEGIN
  day := DVSYS.DBMS_MACUTL.GET_DAY(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Day: '||day);
END;
/
```

Output similar to the following appears:

```
Day: 3
```

GET_MONTH Function

The GET_MONTH function returns the month in Oracle MM (month) format (01–12), in a NUMBER value. It is useful for rule expressions based on time data.

Syntax

```
DVSYS.DBMS_MACUTL.GET_MONTH(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 13–9 GET_MONTH Parameter

Parameter	Description
p_date	Date in MM format (for example, 08 for the month of August). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  month number;
BEGIN
  month := DVSYS.DBMS_MACUTL.GET_MONTH(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Month: '||month);
END;
/
```

Output similar to the following appears:

```
Month: 4
```

GET_YEAR Function

The GET_YEAR function returns the year in Oracle YYYY (year) format (0001–9999), in a NUMBER value. It is useful for rule expressions based on time data.

Syntax

```
DVSYS.DBMS_MACUTL.GET_YEAR(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 13–10 GET_YEAR Parameter

Parameter	Description
p_date	Date in YYYY format (for example, 1984). If you do not specify a date, then Oracle Database Vault uses the SYSDATE function to retrieve the current date and time set for the operating system on which the database resides.

Example

```
SET SERVEROUTPUT ON
DECLARE
  year number;
BEGIN
  year := DVSYS.DBMS_MACUTL.GET_YEAR(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Year: '||year);
END;
/
```

IS_ALPHA Function

The IS_ALPHA function checks whether the character is alphabetic, and then returns a BOOLEAN value. IS_ALPHA returns TRUE if the character is alphabetic.

Syntax

```
DVSYS.DBMS_MACUTL.IS_ALPHA(
  c IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 13–11 IS_ALPHA Parameter

Parameter	Description
c	String with one character

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DVSYS.DBMS_MACUTL.IS_ALPHA('z')
  THEN DBMS_OUTPUT.PUT_LINE('The alphabetic character was found');
 ELSE
  DBMS_OUTPUT.PUT_LINE('No alphabetic characters today.');
 END IF;
END;
```

```
/
```

IS_DIGIT Function

The IS_DIGIT function checks whether the character is numeric, and then returns a BOOLEAN value. IS_DIGIT returns TRUE if the character is a digit.

Syntax

```
DVSYS.DBMS_MACUTL.IS_DIGIT(
  c IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 13–12 IS_DIGIT Parameter

Parameter	Description
c	String with one character

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DVSYS.DBMS_MACUTL.IS_DIGIT('7')
  THEN DBMS_OUTPUT.PUT_LINE('The numeric character was found');
 ELSE
  DBMS_OUTPUT.PUT_LINE('No numeric characters today.');
 END IF;
END;
/
```

IS_DVSYS_OWNER Function

The IS_DVSYS_OWNER function determines whether a user is authorized to manage the Oracle Database Vault configuration, and then returns a BOOLEAN value. IS_DVSYS_OWNER returns TRUE if the user is authorized.

Syntax

```
DVSYS.DBMS_MACUTL.IS_DVSYS_OWNER(
  p_user IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameter

Table 13–13 IS_DVSYS_OWNER Parameter

Parameter	Description
p_user	User to check. To find existing users, query the following views: ▪ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ▪ DVSYS.DBA_REALM_AUTH: Finds the authorization of a particular user or role. See "DBA_DV_REALM_AUTH View" on page 10-22. ▪ DVSYS.DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "DBA_DV_ROLE View" on page 10-23.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DVSYS.DBMS_MACUTL.IS_DVSYS_OWNER('PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('PSMITH is authorized to manage Database Vault.');
 ELSE
  DBMS_OUTPUT.PUT_LINE('PSMITH is not authorized to manage Database Vault.');
 END IF;
END;
/
```

IS_OLS_INSTALLED Function

The IS_OLS_INSTALLED function returns an indicator regarding whether Oracle Label Security is installed, and then returns a TRUE or FALSE BOOLEAN value. If Oracle Label Security is installed, IS_OLS_INSTALLED returns TRUE.

Syntax

```
DVSYS.DBMS_MACUTL.IS_OLS_INSTALLED()
RETURN BOOLEAN;
```

Parameters

None.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DVSYS.DBMS_MACUTL.IS_OLS_INSTALLED()
  THEN DBMS_OUTPUT.PUT_LINE('OLS is installed');
 ELSE
  DBMS_OUTPUT.PUT_LINE('OLS is not installed');
 END IF;
END;
/
```

IS_OLS_INSTALLED_VARCHAR Function

The IS_OLS_INSTALLED_VARCHAR function returns an indicator regarding whether Oracle Label Security is installed, and then returns a Y or N VARCHAR2 value. If Oracle Label Security is installed, thenIS_OLS_INSTALLED_VARCHAR returns Y.

Syntax

```
DVSYS.DBMS_MACUTL.IS_OLS_INSTALLED_VARCHAR()
RETURN VARCHAR2;
```

Parameters

None.

Example

See "IS_OLS_INSTALLED Function" on page 13-13 for an example.

USER_HAS_OBJECT_PRIVILEGE Function

The USER_HAS_OBJECT_PRIVILEGE 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

```
DVSYS.DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE(
  p_user          IN VARCHAR2,
  p_object_owner  IN VARCHAR2,
  p_object_name   IN VARCHAR2,
  p_privilege     IN VARCHAR2)
RETURNS BOOLEAN;
```

Parameters

Table 13–14 USER_HAS_OBJECT_PRIVILEGE Parameters

Parameter	Description
p_user	User or role to check. To find existing users, query the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DBA_ROLES: Finds available roles in the current database instance. See *Oracle Database Reference*. ■ DVSYS.DBA_REALM_AUTH: Finds the authorization of a particular user or role. See "DBA_DV_REALM_AUTH View" on page 10-22. ■ DVSYS.DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "DBA_DV_ROLE View" on page 10-23.
p_object_owner	Object owner. To find the available users, query the DBA_USERS view, described in *Oracle Database Reference*. To find the authorization of a particular user, query the DVSYS.DBA_REALM_AUTH view, described in "DBA_DV_REALM_AUTH View" on page 10-22.
p_object_name	Object name. To find the available objects, query the ALL_OBJECTS view, described in *Oracle Database Reference*. To find objects that are secured by existing realms, query the DVSYS.DBA_DV_REALM_OBJECT view, described in "DBA_DV_REALM_OBJECT View" on page 10-22.
p_privilege	Object privilege (for example, SELECT, UPDATE, INSERT, and so on). To find privileges for a database account excluding PUBLIC privileges, query the DVSYS.DBA_DV_USER_PRIVS view, described in "DBA_DV_USER_PRIVS View" on page 10-25. To find all privileges for a database account, use DVSYS.DBA_DV_USER_PRIVS_ALL, described in "DBA_DV_USER_PRIVS_ALL View" on page 10-26.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DVSYS.DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE(
```

```
   'SECTOR2_APP_MGR', 'OE', 'ORDERS', 'SELECT, UPDATE')
  THEN DBMS_OUTPUT.PUT_LINE('SECTOR2_APP_MGR has privileges.');
   ELSE
  DBMS_OUTPUT.PUT_LINE('SECTOR2_APP_MGR does not have privileges.');
 END IF;
END;
```

USER_HAS_ROLE Function

The USER_HAS_ROLE function checks whether a user has a role privilege, directly or indirectly (through another role), and then returns a BOOLEAN value. If the user has a role privilege, then USER_HAS_ROLE returns TRUE.

Syntax

```
DVSYS.DBMS_MACUTL.USER_HAS_ROLE(
  p_role IN VARCHAR2,
  p_user IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameters

Table 13–15 USER_HAS_ROLE Parameters

Parameter	Description
p_role	Role privilege to check. To find existing roles, query the following views: ■ DBA_ROLES: Finds available roles in the current database instance. See *Oracle Database Reference*. ■ DVSYS.DBA_REALM_AUTH: Finds the authorization of a particular user or role. See "DBA_DV_REALM_AUTH View" on page 10-22. ■ DVSYS.DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "DBA_DV_ROLE View" on page 10-23.
p_user	User to check. To find existing users, query the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DVSYS.DBA_REALM_AUTH: Finds the authorization of a particular user or role. See "Oracle Database Vault Data Dictionary Views" on page 10-11.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DVSYS.DBMS_MACUTL.USER_HAS_ROLE('SECTOR2_APP_MGR', 'PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('User PSMITH has the SECTOR2_APP_MGR role');
   ELSE
  DBMS_OUTPUT.PUT_LINE('User PSMITH does not have the SECTOR2_APP_MGR role.');
 END IF;
END;
/
```

USER_HAS_ROLE_VARCHAR Function

The USER_HAS_ROLE_VARCHAR function checks whether a user has a role privilege, directly or indirectly (through another role), and then returns a VARCHAR2 value. If the user has the role privilege specified, then USER_HAS_ROLE_VARCHAR returns Y.

Syntax

```
DVSYS.DBMS_MACUTL.USER_HAS_ROLE_VARCHAR(
  p_role IN VARCHAR2,
  p_user IN VARCHAR2 DEFAULT USER)
RETURN VARCHAR2;
```

Parameters

Table 13–16 USER_HAS_ROLE_VARCHAR Parameters

Parameter	Description
p_role	Role to check. To find existing roles, query the following views: ■ DBA_ROLES: Finds available roles in the current database instance. See *Oracle Database Reference*. ■ DVSYS.DBA_REALM_AUTH: Finds the authorization of a particular user or role. See "DBA_DV_REALM_AUTH View" on page 10-22. ■ DVSYS.DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "DBA_DV_ROLE View" on page 10-23.
p_user	User to check. To find existing users, query the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DVSYS.DBA_REALM_AUTH: Finds the authorization of a particular user or role. See "DBA_DV_REALM_AUTH View" on page 10-22.

USER_HAS_SYSTEM_PRIVILEGE Function

The USER_HAS_SYSTEM_PRIVILEGE function checks whether a user has a system privilege, directly or indirectly (through a role), and then returns a BOOLEAN value. If the user has the system privilege specified, then USER_HAS_SYSTEM_PRIVILEGE returns TRUE.

Syntax

```
DVSYS.DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE(
  p_privilege  IN VARCHAR2,
  p_user       IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameters

Table 13–17 USER_HAS_SYSTEM_PRIVILEGE Parameters

Parameter	Description
p_privilege	System privilege to check for. To find privileges for a database account excluding PUBLIC privileges, query the DVSYS.DBA_DV_USER_PRIVS view, described in "DBA_DV_USER_PRIVS View" on page 10-25. To find all privileges for a database account, use DVSYS.DBA_DV_USER_PRIVS_ALL, described in "DBA_DV_USER_PRIVS_ALL View" on page 10-26.
p_user	User to check. To find existing users, query the following views: ■ DBA_USERS: Finds available users for the current database instance. See *Oracle Database Reference*. ■ DVSYS.DBA_REALM_AUTH: Finds the authorization of a particular user or role. See "DBA_DV_REALM_AUTH View" on page 10-22.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DVSYS.DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE('EXECUTE', 'PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('User PSMITH has the EXECUTE ANY PRIVILEGE
privilege.');
   ELSE
  DBMS_OUTPUT.PUT_LINE('User PSMITH does not have the EXECUTE ANY PRIVILEGE
privilege.');
 END IF;
END;
/
```

14

Using the Oracle Database Vault PL/SQL Interfaces

This chapter contains:

- 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 the `EXECUTE` privilege on all `DVSYS` package procedures to the general database account population. The procedures and functions expose only the minimum methods that are required. All of these functions and procedures are publicly available for applications that need them.

Table 14–1 lists the default run-time PL/SQL procedures and functions.

Table 14–1 DVSYS Functions

Procedure or Function	Parameter
SET_FACTOR Procedure	Sets a factor
GET_FACTOR Function	Retrieves a factor
GET_TRUST_LEVEL Function	Retrieves the trust level assigned to a factor
GET_TRUST_LEVEL_FOR_IDENTITY Function	Retrieves the trust level for a specified factor and an identity
ROLE_IS_ENABLED Function	Checks whether the specified database role is enabled
GET_FACTOR_LABEL Function	Retrieves the label for the specified factor when the factor has a label assigned to it for the specified Oracle Label Security policy

SET_FACTOR Procedure

The SET_FACTOR procedure can be exposed to an application that requires the ability to set factor identities dynamically. It wraps the package procedure DVSYS.DBMS_MACADM.SET_FACTOR. When a factor has a rule set associated with it for assignment and if the rule set returns true, then the value is set. Normal rule set handling occurs, and the factor value (identity) validation method is called. This procedure is available (to execute) to the general database account population.

Syntax

```
DVSYS.SET_FACTOR(
  p_factor IN VARCHAR2,
  p_value  IN VARCHAR2);
```

Parameters

Table 14–2 SET_FACTOR Parameters

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.
p_value	Identity value, up to 1024 characters in mixed case. To find the identities for each factor in the current database instance, query the DVSYS.DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17.

Example

```
EXECUTE DVSYS.SET_FACTOR(''Sector2_ClientID'', ''identity'');
```

GET_FACTOR Function

The GET_FACTOR function is exposed to the DVF schema to allow the public factor functions to resolve the identity of a factor. This enables the F$ functions in the DVF schema. This function is available (to execute) to the general database account population.

Syntax

```
DVSYS.GET_FACTOR(
  p_factor IN VARCHAR2)
RETURN VARCHAR2;
```

Parameter

Table 14–3 GET_FACTOR Parameter

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Factor Identity',
  rule_expr => 'DVSYS.GET_FACTOR(''Sector2_ClientID'')');
END;
/
```

GET_TRUST_LEVEL Function

The GET_TRUST_LEVEL function returns the trust level of the current session identity for the factor requested. This function is available (to execute) to the general database account population. See "Creating and Configuring a Factor Identity" on page 7-11 for a listing of the available trust levels.

Syntax

```
DVSYS.GET_TRUST_LEVEL(
  p_factor IN VARCHAR2)
RETURN VARCHAR2;
```

Parameter

Table 14–4 GET_TRUST_LEVEL Parameter

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Trust Level',
  rule_expr => 'DVSYS.GET_TRUST_LEVEL(''Sector2_ClientID'')');
END;
/
```

GET_TRUST_LEVEL_FOR_IDENTITY Function

The GET_TRUST_LEVEL_FOR_IDENTITY function returns the trust level for the factor and identity requested. This function is available (to execute) to the general database account population. See "Creating and Configuring a Factor Identity" on page 7-11 for a listing of the available trust levels.

Syntax

```
DVSYS.GET_TRUST_LEVEL_FOR_IDENTITY(
  p_factor   IN VARCHAR2,
  p_identity IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 14–5 GET_TRUST_LEVEL_FOR_IDENTITY Parameters

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view, described in "DBA_DV_FACTOR View" on page 10-15.
p_identity	Identity value. To find the identities for each factor in the current database instance, query the DBA_DV_IDENTITY view, described in "DBA_DV_IDENTITY View" on page 10-17.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Identity Trust Level',
  rule_expr => 'DVSYS.GET_TRUST_LEVEL_FOR_IDENTITY(''Sector2_ClientID'',
''identity'')');
END;
/
```

ROLE_IS_ENABLED Function

The ROLE_IS_ENABLED function returns a boolean value that specifies whether a database role has been enabled. This function is available (to execute) to the general database account population.

Syntax

```
DVSYS.ROLE_IS_ENABLED(
  p_role IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 14–6 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*. ■ DVSYS.DBA_DV_REALM_AUTH: Finds the authorization of a particular role. See "DBA_DV_REALM View" on page 10-21. ■ DVSYS.DBA_DV_ROLE: Finds existing secure application roles used in privilege management. See "DBA_DV_ROLE View" on page 10-23.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check if SYSADM Role Is Enabled',
  rule_expr => 'DVSYS.ROLE_IS_ENABLED(''SYSADM'')');
END;
/
```

GET_FACTOR_LABEL Function

The GET_FACTOR_LABEL function returns the label for the specified factor when the factor has a label assigned to it for the specified Oracle Label Security policy. The 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 7-12 for more information about factor labels.

Syntax

```
DVSYS.GET_FACTOR_LABEL(
  p_factor      IN VARCHAR2,
  p_policy_name IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 14–7 GET_FACTOR_LABEL Parameters

Parameter	Description
p_factor	Factor name. To find the available factors in the current database instance, query the DVSYS.DBA_DV_FACTOR view. To find factors that are associated with Oracle Label Security policies, use DVSYS.DBA_DV_MAC_POLICY_FACTOR. See "DBA_DV_FACTOR View" on page 15 and "DBA_DV_MAC_POLICY_FACTOR View" on page 10-19.
p_policy_name	Oracle Label Security policy name. Use the following views to find information about policies and factors in the current database instance: ■ DVSYS.DBA_DV_MAC_POLICY: Lists Oracle Label Security policies defined in the current database instance. See "DBA_DV_MAC_POLICY View" on page 10-19. ■ DVSYS.DBA_DV_MAC_POLICY_FACTOR: Lists the factors that are associated with Oracle Label Security policies for the current database instance. See "DBA_DV_MAC_POLICY_FACTOR View" on page 10-19. ■ DVSYS.DBA_DV_POLICY_LABEL: Lists the Oracle Label Security label for each factor identifier in the DVSYS.DBA_DV_IDENTITY view for each policy. See "DBA_DV_POLICY_LABEL View" on page 10-20.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get the ClientID Factor Label',
  rule_expr => 'DVSYS.GET_FACTOR_LABEL(''Sector2_ClientID'', ''Access
Locations'')');
END;
/
```

Oracle Database Vault 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. Oracle Database Vault maintains these functions when you use the DVSYS.DBMS_MACADM

PL/SQL package to manage the various factors. The functions are then available to the general database account population through PL/SQL functions and standard SQL. This enables factors to be used in Oracle Label Security, Oracle Virtual Private Database (VPD), and so on.

Typically, you can incorporate these functions into rule expressions. For example:

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
 rule_name => 'Not Internal DBA',
 rule_expr => 'DVF.F$SESSION_USER NOT IN (''JSMTIH'', ''TBROWN'')');
END;
/
```

To find the value of a factor function, select from the `DUAL` system table. For example:

```
SELECT DVF.F$SESSION_USER FROM DUAL;

F$SESSION_USER
-----------------------------------------------
LBROWN_DVOWNER
```

The name of the factor itself is case-insensitive. For example, the following statements return the same result

```
select dvf.f$session_user from dual;

SELECT DVF.F$SESSION_USER FROM DUAL;
```

Table 14–8 lists the default factor functions.

Table 14–8 Installed Oracle Database Vault Factor Functions

DVF Factor Function	Description
F$AUTHENTICATION_METHOD Function	Returns the method of authentication in `VARCHAR2` data type. In the list that follows, the type of user is followed by the method returned
F$CLIENT_IP Function	Returns the IP address of the computer from which the client is connected
F$DATABASE_DOMAIN Function	Returns the domain of the database as specified in the `DB_DOMAIN` initialization parameter
F$DATABASE_HOSTNAME Function	Returns the host name of the computer on which the database instance is running
F$DATABASE_INSTANCE Function	Returns the database instance identification number of the current database instance
F$DATABASE_IP Function	Returns the IP address of the computer on which the database instance is running
F$DATABASE_NAME Function	Returns the name of the database as specified in the `DB_NAME` initialization parameter
F$DOMAIN Function	Returns a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level
F$ENTERPRISE_IDENTITY Function	Returns the enterprise-wide identity for a user
F$IDENTIFICATION_TYPE Function	Returns the way the schema of a user was created in the database. Specifically, it reflects the `IDENTIFIED` clause in the `CREATE USER` or `ALTER USER` syntax.

Table 14–8 (Cont.) Installed Oracle Database Vault Factor Functions

DVF Factor Function	Description
F$LANG Function	Returns the ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter
F$LANGUAGE Function	Returns the language and territory currently used by your session, in VARCHAR2 data type, along with the database character set
F$MACHINE Function	Returns the computer (host) name for the database client that established the database session
F$NETWORK_PROTOCOL Function	Returns the network protocol being used for communication, as specified in the PROTOCOL=*protocol* portion of the connect string
F$PROXY_ENTERPRISE_IDENTITY Function	Returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user
F$SESSION_USER Function	Returns the database user name by which the current user is authenticated

F$AUTHENTICATION_METHOD Function

The F$AUTHENTICATION_METHOD function returns the method of authentication in VARCHAR2 data type. In the list that follows, the type of user is followed by the method returned:

- Password-authenticated enterprise user, local database user, or SYSDBA/SYSOPER using Password File; proxy with user name using password: PASSWORD
- Kerberos-authenticated enterprise or external user: KERBEROS
- SSL-authenticated enterprise or external user: SSL
- Radius-authenticated external user: RADIUS
- Operating system-authenticated external user or SYSDBA/SYSOPER: OS
- DCE-authenticated external user: DCE
- Proxy with certificate, distinguished name (DN), or user name without using password: NONE

You can use IDENTIFICATION_TYPE to distinguish between external and enterprise users when the authentication method is Password, Kerberos, or SSL.

Syntax

```
DVF.F$AUTHENTICATION_METHOD ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check SSL Authentication Method',
  rule_expr => 'DVF.F$AUTHENTICATION_METHOD = ''SSL''');
END;
/
```

F$CLIENT_IP Function

The F$CLIENT_IP function returns the IP address of the computer from which the client is connected, in VARCHAR2 data type.

Syntax

```
DVF.F$CLIENT_IP ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client IP Address',
  rule_expr => 'DVF.F$CLIENT_IP BETWEEN ''192.0.2.10'' AND ''192.0.2.20''');
END;
/
```

F$DATABASE_DOMAIN Function

The F$DATABASE_DOMAIN function returns the domain of the database as specified in the DB_DOMAIN initialization parameter, in VARCHAR2 data type.

Syntax

```
DVF.F$DATABASE_DOMAIN ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client Database Domain',
  rule_expr => 'DVF.F$DATABASE_DOMAIN NOT IN (''EXAMPLE'', ''YOURDOMAIN'')');
END;
/
```

F$DATABASE_HOSTNAME Function

The F$DATABASE_HOSTNAME function returns the host name of the computer on which the instance is running, in VARCHAR2 data type.

Syntax

```
DVF.F$DATABASE_HOSTNAME ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
```

```
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Host Name',
  rule_expr => 'DVF.F$DATABASE_HOSTNAME IN (''SHOBEEN'', ''MAU'')');
END;
/
```

F$DATABASE_INSTANCE Function

The F$DATABASE_INSTANCE function returns the instance identification number of the current database instance, in VARCHAR2 data type.

Syntax

```
DVF.F$DATABASE_INSTANCE ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database Instance ID',
  rule_expr => 'DVF.F$DATABASE_INSTANCE = ''SALES_DB''');
END;
/
```

F$DATABASE_IP Function

The F$DATABASE_IP function returns the IP address of the computer on which the database instance is running, in VARCHAR2 data type.

Syntax

```
DVF.F$DATABASE_IP ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database IP address',
  rule_expr => 'DVF.F$DATABASE_IP = ''192.0.2.5''');
END;
/
```

F$DATABASE_NAME Function

The F$DATABASE_NAME function returns the name of the database as specified in the DB_NAME initialization parameter, in VARCHAR2 data type.

Syntax

```
DVF.F$DATABASE_NAME ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database DB_NAME Name',
  rule_expr => 'DVF.F$DATABASE_NAME = ''ORCL''');
END;
/
```

F$DOMAIN Function

The `F$DOMAIN` function returns a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level. The return type is `VARCHAR2`.

You can identify a domain using factors such as host name, IP address, and database instance names of the Oracle Database Vault nodes in a secure access path to the database. Each domain can be uniquely determined using a combination of the factor identifiers that identify the domain. You can use these identifying factors and possibly additional factors to define the Maximum Security Label within the domain. This restricts data access and commands, depending on the physical factors about the Oracle Database Vault session. Example domains of interest may be Corporate Sensitive, Internal Public, Partners, and Customers.

Syntax

```
DVF.F$DOMAIN ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Domain',
  rule_expr => 'DVF.F$DOMAIN = ''EXAMPLE.COM''');
END;
/
```

F$ENTERPRISE_IDENTITY Function

The `F$ENTERPRISE_IDENTITY` function returns the enterprise-wide identity for a user, in `VARCHAR2` data type:

- For enterprise users: the Oracle Internet Directory DN.
- For external users: the external identity (Kerberos principal name, Radius and DCE schema names, operating system user name, certificate DN).
- For local users and `SYSDBA`/`SYSOPER` logins: NULL.

The value of the attribute differs by proxy method:

- For a proxy with DN: the Oracle Internet Directory DN of the client.

- For a proxy with certificate: the certificate DN of the client for external users; the Oracle Internet Directory DN for global users.
- For a proxy with user name: the Oracle Internet Directory DN if the client is an enterprise user; NULL if the client is a local database user.

Syntax

```
DVF.F$ENTERPRISE_IDENTITY ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check User Enterprise Identity',
  rule_expr => 'DVF.F$ENTERPRISE_IDENTITY NOT IN (''JSMITH'', ''TSMITH'')');
END;
/
```

F$IDENTIFICATION_TYPE Function

The `F$IDENTIFICATION_TYPE` function returns the way the schema of a user was created in the database. Specifically, it reflects the `IDENTIFIED` clause in the `CREATE/ALTER USER` syntax. The return type is `VARCHAR2`. In the list that follows, the syntax used during schema creation is followed by the identification type returned:

- `IDENTIFIED BY` *`password`*`: LOCAL`
- `IDENTIFIED EXTERNALLY: EXTERNAL`
- `IDENTIFIED GLOBALLY: GLOBAL SHARED`
- `IDENTIFIED GLOBALLY AS DN: GLOBAL PRIVATE`

Syntax

```
DVF.F$IDENTIFICATION_TYPE ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check User Schema Creation Type',
  rule_expr => 'DVF.F$IDENTIFICATION_TYPE = ''GLOBAL SHARED''');
END;
/
```

F$LANG Function

The `F$LANG` function returns the ISO abbreviation for the language name, a shorter form than the existing `LANGUAGE` parameter, for the session of the user. The return type is `VARCHAR2`.

See *Oracle Database Globalization Support Guide* for a listing of supported languages for Oracle Database.

Syntax

```
DVF.F$LANG ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check ISO Abbreviated Language Name',
  rule_expr => 'DVF.F$LANG IN (''EN'', ''DE'', ''FR'')');
END;
/
```

F$LANGUAGE Function

The `F$LANGUAGE` function returns the language and territory currently used by a user session, in `VARCHAR2` data type, along with the database character set, in the following form:

`language_territory.characterset`

See *Oracle Database Globalization Support Guide* for a listing of supported languages and territories for Oracle Database.

Syntax

```
DVF.F$LANGUAGE ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Session Language and Territory',
  rule_expr => 'DVF.F$LANGUAGE = ''AMERICAN_AMERICA.WE8ISO8859P1''');
END;
/
```

F$MACHINE Function

The `F$MACHINE` function returns the computer (host) name for the database client that established the database session. The return type is `VARCHAR2`.

Syntax

```
DVF.F$MACHINE ()
RETURN VARCHAR2;
```

Parameter

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client Computer Host Name',
  rule_expr => 'DVF.F$MACHINE NOT IN (''SHOBEEN'', ''SEBASTIAN'')');
END;
/
```

F$NETWORK_PROTOCOL Function

The F$NETWORK_PROTOCOL function returns the network protocol being used for communication, as specified in the PROTOCOL=*protocol* portion of the connect string. The return type is VARCHAR2.

Syntax

```
DVF.F$NETWORK_PROTOCOL ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Network Protocol',
  rule_expr => 'DVF.F$NETWORK_PROTOCOL = ''TCP''');
END;
/
```

F$PROXY_ENTERPRISE_IDENTITY Function

The F$PROXY_ENTERPRISE_IDENTITY function returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user. The return type is VARCHAR2.

Syntax

```
DVF.F$PROXY_ENTERPRISE_IDENTITY ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get OID DN of Enterprise User',
  rule_expr => 'DVF.F$PROXY_ENTERPRISE_IDENTITY = ''cn=Provisioning Admins''');
END;
/
```

F$SESSION_USER Function

The F$SESSION_USER function returns the database user name by which the current user is authenticated. This value remains the same throughout the session. The return type is VARCHAR2.

Syntax

```
DVF.F$SESSION_USER ()
RETURN VARCHAR2;
```

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database User Name',
  rule_expr => 'DVF.F$SESSION_USER IN (''JSMITH'', ''TSMITH'')');
END;
/
```

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 14–9 lists the default rule functions.

Table 14–9 Installed Oracle Database Vault PL/SQL Rule Set Functions

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

DV_SYSEVENT Function

The `DV_SYSEVENT` function returns the system event firing the rule set, in `VARCHAR2` data type. The event name is the same as that in the syntax of the SQL statement (for example, `INSERT`, `CREATE`).

Syntax

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

Parameters

None.

Example

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

DV_LOGIN_USER Function

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

Syntax

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

Parameters

None.

Example

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

DV_INSTANCE_NUM Function

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

Syntax

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

Parameters

None.

Example

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

DV_DATABASE_NAME Function

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

Syntax

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

Parameters

None.

Example

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

DV_DICT_OBJ_TYPE Function

The DV_DICT_OBJ_TYPE function returns the type of the dictionary object on which the database operation occurred (for example, table, procedure, or view). The return type is VARCHAR2.

Syntax

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

Parameters

None.

Example

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

DV_DICT_OBJ_OWNER Function

The DV_DICT_OBJ_OWNER function returns the name of the owner of the dictionary object on which the database operation occurred. The return type is VARCHAR2.

Syntax

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

Parameters

None.

Example

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

DV_DICT_OBJ_NAME Function

The DV_DICT_OBJ_NAME function returns the name of the dictionary object on which the database operation occurred. The return type is VARCHAR2.

Syntax

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

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Dictionary Object Name',
  rule_expr => 'DVSYS.DV_DICT_OBJ_NAME = ''SALES''');
END;
/
```

DV_SQL_TEXT Function

The DV_SQL_TEXT function returns the first 4000 characters of SQL text of the database statement used in the operation The return type is VARCHAR2.

Syntax

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

Parameters

None.

Example

```
BEGIN
 DVSYS.DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check SQL Text',
  rule_expr => 'DVSYS.DV_SQL_TEXT = ''SELECT SALARY FROM HR.EMPLOYEES''');
END;
/
```

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 14–10 lists these packages. Chapter 11, "Using the DVSYS.DBMS_MACADM Package" describes these packages in detail.

Table 14–10 Oracle Database Vault Administrator and Run-Time PL/SQL Packages

Package	Description
`DVSYS.DBMS_MACADM`	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 Chapter 11, "Using the DVSYS.DBMS_MACADM Package" for more information.
`DVSYS.DBMS_MACSEC_ROLES`	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 Chapter 12, "Using the DVSYS.DBMS_MACSEC_ROLES Package" for more information.
`DVSYS.DBMS_MACUTL`	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 Chapter 13, "Using the 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`

15

Monitoring Oracle Database Vault

This chapter contains:

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

> **Note:** To make the charts used in the Monitor page accessible for to users of assistive technology, see "Enabling Oracle Database Vault Accessibility" in *Oracle Database Vault Installation Guide*.

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.

To check for security violations:

1. Log in to Oracle Database Vault Administrator as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

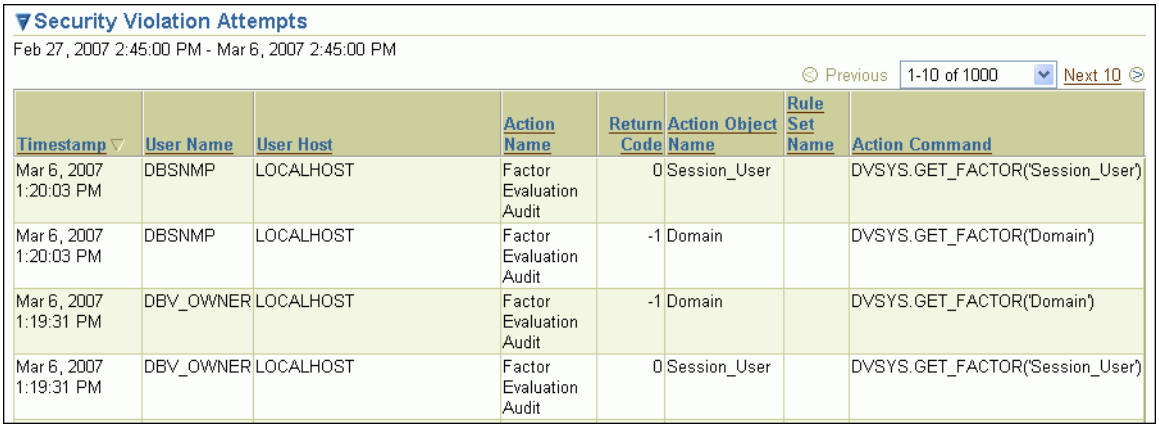
   "Starting Oracle Database Vault" on page 3-1 explains how to log in.

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

Previous 1-10 of 1000 Next 10

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 as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

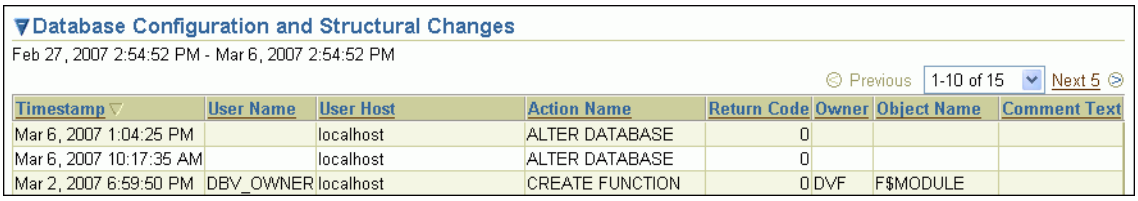
   "Starting Oracle Database Vault" on page 3-1 explains how to log in.

2. In the Administration page, click **Monitor**.
3. At the top of the Monitor page, set a period 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

Previous 1-10 of 15 Next 5

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.
- **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 as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

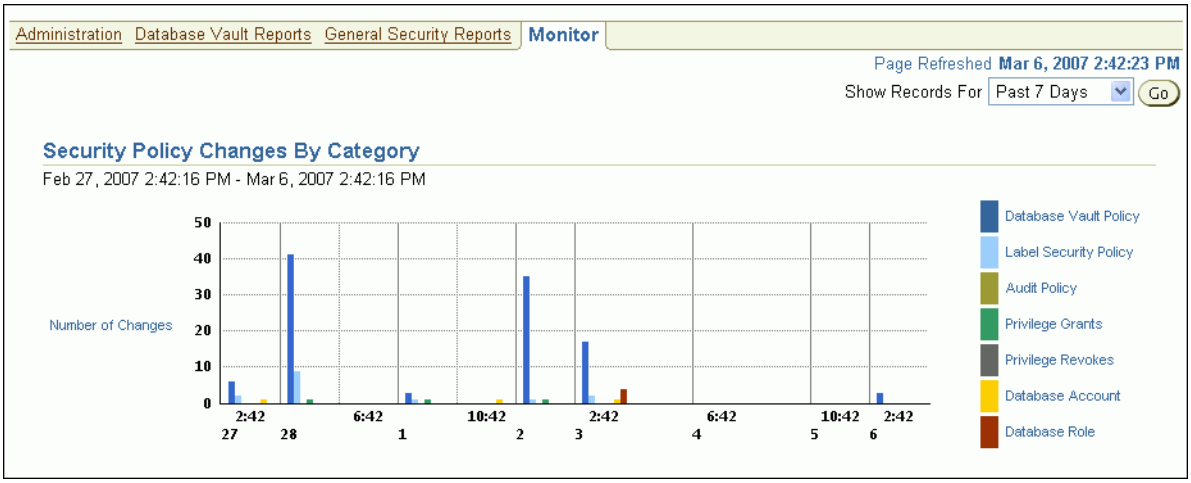
   "Starting Oracle Database Vault" on page 3-1 explains how to log on.

2. In the Administration page, click **Monitor**.
3. At the top of the Monitor page, set a period 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.



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 as a user who has been granted `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

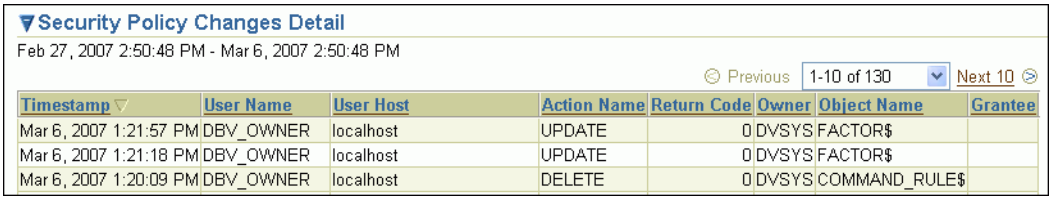
   "Starting Oracle Database Vault" on page 3-1 explains how to log in.

2. In the Administration page, click **Monitor**.
3. At the top of the Monitor page, set a period 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

Previous 1-10 of 130 Next 10

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$	

16

Oracle Database Vault Reports

This chapter contains:

- 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 10-4
- "Oracle Database Vault Configuration Administrator Role, DV_ADMIN" on page 10-6
- "Oracle Database Vault Security Analyst Role, DV_SECANALYST" on page 10-8

How to Run Oracle Database Vault Reports

To run Oracle Database Vault reports:

1. Log in to Database Vault Administrator a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` (least privileged) role.

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

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 16-2
   - "Generating General Security Reports" on page 16-5

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 and 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 when 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:

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

Oracle Database Vault Auditing Reports

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	Instance Number
Timestamp	Object Name
Return Code	Rule Set
Account	Command
User Host	

Secure Application Role Audit Report

The Secure Application Role Audit Report shows the audit records generated by the secure application role-enabling operation for Oracle Database Vault.

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	LBACSYS	SYS	WMSYS
DMSYS	MDSYS	SYSMAN	
DVSYS	ORDSYS	SYSTEM	
EXFSYS	PUBLIC	WKSYS	

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 and the source object name and type. This report shows where the potentially sensitive object is being used. By looking at several accounts, you might be able to see patterns that can help you develop restricted roles. These restricted roles can replace PUBLIC grants on widely used sensitive objects.

Database Account System Privileges Reports

The database account system privileges reports 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 the `EXECUTE` privilege on system packages that can be used to access operating system resources or other powerful system packages. The following system PL/SQL packages are included:

`DBMS_ALERT`	`DBMS_RANDOM`
`DBMS_BACKUP_RESTORE`	`DBMS_REPAIR`
`DBMS_CAPTURE_ADM`	`DBMS_REPCAT`
`DBMS_DDL`	`DBMS_REPCAT_ADMIN`

DBMS_DISTRIBUTED_TRUST_ADMIN
DBMS_FGA
DBMS_JOB
DBMS_LDAP
DBMS_LOB
DBMS_LOGMNR
DBMS_LOGMNR_D
DBMS_OBFUSCATION_TOOLKIT
DBMS_ORACLE_TRACE_AGENT
DBMS_PIPE
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 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

> **See Also:** "DBA_DV_PUB_PRIVS View" on page 10-20 to find the values on which the counts listed in these reports are based

Privileges Distribution By Grantee Report

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

See Also:

- "DBA_DV_PUB_PRIVS View" on page 10-20
- "DBA_DV_USER_PRIVS View" on page 10-25
- "DBA_DV_USER_PRIVS_ALL View" on page 10-26

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 C, "Oracle Database Vault Security Guidelines".

Security Policy Exemption Report

The Security Policy Exemption Report shows database (but not Oracle Database Vault) accounts and roles that have the `EXEMPT ACCESS POLICY` system privilege granted to them. Accounts that have this privilege can bypass all Virtual Private Database (VPD) policy filters and any Oracle Label Security policies that use Oracle Virtual Private Database indirectly. This is a powerful system privilege that should be granted only if absolutely necessary, as it presents a target to gain access to sensitive information in tables that are protected by Oracle Virtual Private Database or Oracle Label Security. You can use the auditing policies described in Appendix A, "Auditing Oracle Database Vault" to audit the use of this privilege.

BECOME USER Report

The BECOME USER Report shows all database accounts roles that have the `BECOME USER` system privilege. This is a very powerful system privilege: it enables the `IMP_FULL_DATABASE` and `EXP_FULL_DATABASE` roles for use with Oracle Data Pump. Accounts that possess this privilege can be misused to get sensitive information or to compromise an application.

ALTER SYSTEM or ALTER SESSION Report

The ALTER SYSTEM or ALTER SESSION Report shows all database accounts and roles that have the `ALTER SYSTEM` or `ALTER SESSION` privilege. Oracle recommends that you restrict these privileges only to those accounts and roles that truly need them (for example, the `SYS` account and the `DV_ADMIN` role). The `ALTER SYSTEM` statement can be used to change the security-related database initialization parameters that are set to recommended values as part of the Oracle Database Vault security strengthening service. Both the `ALTER SYSTEM` and `ALTER SESSION` statements can be used to dump database trace files, potentially containing sensitive configuration information, to the operating system.

For guidelines on using the `ALTER SYSTEM` and `ALTER SESSION` privileges, see "Security Considerations for the ALTER SYSTEM and ALTER SESSION Privileges" on page C-10.

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 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 Appendix A, "Auditing Oracle Database Vault,", and any auditing records that are generated for audit statements you have defined.

This report only displays audit records that are captured if the database initialization parameter `AUDIT_TRAIL` has been set to `DB`. For more information about the `AUDIT_TRAIL` parameter, see *Oracle Database SQL 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 they are available to `PUBLIC`, and what their privileges are. Directory objects should exist only for secured operating system (OS) directories, and access to them within the database should be protected. You should never use the root operating system directory on any storage device (for example, `/`) because it allows remote database sessions to look at all files on the device.

Objects Dependent on Dynamic SQL Report

The Objects Dependent on Dynamic SQL Report shows objects that leverage dynamic SQL. Potential intruders have a greater chance of using this channel if parameter checking or bind variables are not used. The report helps by narrowing the scope of where to look for problems by pointing out who is using dynamic SQL. Such objects can be a target for a SQL injection attack and must be secured to avoid this type of attack. After determining the objects that use dynamic SQL, do the following:

- Check the privileges that client applications (for example, a Web application) have over the object.
- Check the access granted for the object to `PUBLIC` or a wider account base.
- Validate parameters.
- Use bind variables where possible.

Unwrapped PL/SQL Package Bodies Report

The Unwrapped PL/SQL Package Bodies Report displays PL/SQL package procedures that are not wrapped. Oracle provides a wrap utility that obfuscates code to the point where it cannot be read in the data dictionary or from the data dictionary

views. This helps reduce the ability of an intruder to circumvent data protection by eliminating the ability to read source code that manipulates data.

Username/Password Tables Report

The Username/Password Tables Report helps to identify application tables in the database that store user names and password strings. You should examine these tables to determine if the information is encrypted. (Search for column names such as `%USER%NAME%` or `%PASSWORD%`.) If it is not, modify the code and applications using these tables to protect them from being visible to database sessions.

Tablespace Quotas Report

The Tablespace Quotas Report shows all database accounts that have quotas on one or more tablespaces. These tablespaces can become potential targets for denial-of-service (DoS) attacks.

Non-Owner Object Trigger Report

The Non-Owner Object Trigger Report lists triggers that are owned by a database account that is different from the account that owns the database object on which the trigger acts. If the trigger is not part of a trusted database application, then it can *steal* sensitive data, possibly from tables protected through Oracle Label Security or Virtual Private Database (VPD), and place it into an unprotected table for subsequent viewing or export.

A

Auditing Oracle Database Vault

This appendix contains:

- Oracle Database Vault Specific Audit Events
- Archiving and Purging the Oracle Database Vault Audit Trail
- Oracle Database Audit Settings Created for Oracle Database Vault

Oracle Database Vault Specific Audit Events

This section contains:

- Oracle Database Vault Audit Events
- Format of the Oracle Database Vault Audit Trail

Oracle Database Vault Audit Events

Oracle Database Vault uses audit events to track the following activities:

- **Realm Audit.** You can audit both successful and failed actions, based on the auditing option that you set when you created the realm. The exception to this is actions performed by the schema owner.
- **Rule Set Audit.** Audits the rule set processing results. You can audit both successful and failed processing. Realm authorizations can be managed using rule sets. You can audit the rule set processing results. Factor assignments and secure application roles audits can be managed using a rule set.
- **Factor Audit.** You can audit both successful and failed factor processing. For failed factor processing, you can audit on all or any of the following events: Retrieval Error, Retrieval Null, Validation Error, Validation False, Trust Level Null, or Trust Level Less Than Zero.
- **Oracle Label Security Session Initialization Failed.** Audits instances where the Oracle Label Security session fails to initialize.
- **Oracle Label Security Attempt to Upgrade Session Label Failed.** Audits instances where the Oracle Label Security component prevents a session from setting a label that exceeds the maximum session label.

See Also:

- "Audit Options" on page 7-9 (for factors)
- "Audit Options" on page 5-3 (for rule sets)
- "Defining Realm Authorization" on page 4-5
- Chapter 16, "Oracle Database Vault Reports" for information about viewing the audit reports

Format of the Oracle Database Vault Audit Trail

The Oracle Database Vault 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 Oracle Database audit trail, and how auditing is enabled in the database has no effect how Oracle Database Vault collects its audit data in the AUDIT_TRAIL$ table. In fact, even if auditing has been disabled in Oracle Database, then the Oracle Database Vault audit functionality continues to write to the AUDIT_TRAIL$ table. Users who have been granted the DV_OWNER, DV_ADMIN, or DV_SECANALYST role can directly query the DVYS.AUDIT_TRAIL$ table.

Note: Oracle Audit Vault can collect the audit data for Oracle Database Vault. See *Oracle Audit Vault Administrator's Guide* for more information.

Table A–1 describes the format of the audit trail, which you must understand if you plan to create custom reports that use the AUDIT_TRAIL$ table.

Table A–1 Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR2(255)		Operating system login user name of the user whose actions were audited
USERNAME	VARCHAR2(30)		Name of the database user whose actions were audited
USERHOST	VARCHAR2(128)		Client computer name
TERMINAL	VARCHAR2(255)		Identifier for the user's terminal
TIMESTAMP	DATE		Date and time of creation of the audit trail entry (in the local database session time zone)
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–1 (Cont.) Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
ACTION	NUMBER	NOT NULL	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
ACTION_OBJECT_ID	NUMBER		The unique identifier of the record in the table specified under OBJ_NAME.
ACTION_OBJECT_NAME	VARCHAR2(128)		The unique name or natural key of the record in the table specified under OBJ_NAME
ACTION_COMMAND	VARCHAR2(4000)		The SQL text of the command procedure that was executed that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)		The labels for all audit options specified in the record that resulted in the audit event being triggered. For example, a factor set operation that is supposed to audit on get failure and get NULL would indicate these two options.
RULE_SET_ID	NUMBER		The unique identifier of the rule set that was executing and caused the audit event to trigger
RULE_SET_NAME	VARCHAR2(30)		The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER		Not used
RULE_NAME	VARCHAR2(30)		Not used
FACTOR_CONTEXT	VARCHAR2(4000)		An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered
COMMENT_TEXT	VARCHAR2(4000)		Text comment on the audit trail entry, providing more information about the statement audited
SESSIONID	NUMBER	NOT NULL	Numeric identifier for each Oracle session
ENTRYID	NUMBER	NOT NULL	Same as the value in the ID# column

Table A–1 (Cont.) Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
STATEMENTID	NUMBER	NOT NULL	Numeric identifier for the statement invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
RETURNCODE	NUMBER	NOT NULL	Oracle error code generated by the action. The error code for a statement or procedure invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
EXTENDED_TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE		Time stamp of creation of the audit trail entry (time stamp of user login for entries) in UTC (Coordinated Universal Time) time zone
PROXY_SESSIONID	NUMBER		Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)		Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER		Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)		Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(30)		Database login user name of the user whose actions were audited
CREATE_DATE	DATE		Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(30)		Same as CREATED_BY column value
UPDATE_DATE	DATE		Same as UPDATED_BY column value

Archiving and Purging the Oracle Database Vault Audit Trail

You can create an archive of the Oracle Database Vault audit trail by exporting the AUDIT_TRAIL$ system table, which is owned by DVSYS, to a dump file. You should periodically archive and then purge the audit trail to prevent it from growing too large.

To archive and purge 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:

   ```
   @/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 must have this account unlocked when you run the EXPDP export command. Ensure that you are still logged on as SYS, connecting AS SYSDBA.

   ```
   SQL> ALTER USER DVSYS IDENTIFIED BY password ACCOUNT UNLOCK;
   ```

4. At the operating system command prompt, create a directory for the Oracle Database Vault audit trail (for example, in $ORACLE_BASE/admin/$DB_UNIQUE_NAME/dvaudit).

   You may want to keep this directory in the same location as the operating system audit trail directories for Oracle Database, which by default is in the $ORACLE_BASE/admin/$DB_UNIQUE_NAME/adump directory. In SQL*Plus, you can check the location of the audit trail directory as follows:

   ```
   SQL> SHOW PARAMETER AUDIT_FILE_DEST

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

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 ANY DIRECTORY privilege.

   For example:

   ```
   SQL> CREATE DIRECTORY dv_audit_dir AS
   '/opt/oracle/app/oracle/admin/orcl/dvaudit';
   ```

   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:

   ```
   SQL> GRANT READ, WRITE ON DIRECTORY dvaudit TO dvsys;
   ```

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

   ```
   EXPDP DVSYS
   Enter password: password
   DIRECTORY=dvaudit \
   TABLES=DVSYS.AUDIT_TRAIL$ \
   QUERY=DVSYS.AUDIT_TRAIL$:"WHERE timestamp < 2009-08-03:19:34:59"
   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 2009-08-03:19:34:59.

- 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. In SQL*Plus, 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 2009-08-03:19:34:59, enter the following statement:

   ```
   SQL> DELETE FROM DVSYS.AUDIT_TRAIL$ WHERE TIMESTAMP < 2009-08-03:19:34:59;
   ```

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

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

9. Exit SQL*Plus.
10. Re-enable Oracle Database Vault.

    See "Step 3: Enable Oracle Database Vault" on page B-5 for more information.

Oracle Database Audit Settings Created for Oracle Database Vault

When you install Oracle Database Vault, it creates a number of AUDIT settings in the database. However, in order for these audit settings to take place, auditing must be enabled in this database. You can check if auditing is enabled by using the SHOW PARAMETER command to find the value of the AUDIT_TRAIL initialization parameter. By default, auditing is disabled in Oracle Database.

If the AUDIT_TRAIL parameter is set to NONE, then auditing is not enabled, so you must set AUDIT_TRAIL. For detailed information about the AUDIT_TRAIL parameter, see *Oracle Database Security Guide* and *Oracle Database Reference*.

Table A–2 lists the AUDIT settings that Oracle Database Vault adds to the database.

Table A–2 Audit Policy Settings Oracle Database Vault Adds to Oracle Database

Audit Setting Type	Audited Statements (BY ACCESS and on Success or Failure Unless Otherwise Noted)
System Audit Settings/System Privilege Usage	ALTER ANY
	CREATE ANY
	DELETE ANY
	DROP ANY
	EXECUTE ANY (WHENEVER NOT SUCCESSFUL)
	FORCE ANY
	GRANT ANY
	INSERT ANY
	UPDATE ANY
System Audit Settings/Object Management	ALTER DATABASE, PROFILE, ROLLBACK SEGMENT, SESSION, SYSTEM, TABLE, TABLESPACE, USER
	CREATE CLUSTER, DATABASE LINK, INDEXTYPE, LIBRARY, OPERATOR, PUBLIC SYNONYM, PROCEDURE, PROFILE, ROLE, ROLLBACK SEGMENT, SEQUENCE, SESSION, SNAPSHOT, SYNONYM, TABLE, TABLESPACE, TRIGGER, TYPE, USER, VIEW
	TRUNCATE
System Audit Settings/Intrusive Commands	ALTER SESSION
	BECOME USER
	CREATE SESSION
	DEBUG CONNECT SESSION
	RESTRICTED SESSION
System Audit Settings/Administration Commands	ADMINISTER DATABASE TRIGGER
	BACKUP ANY TABLE
	EXEMPT ACCESS POLICY
	MANAGE TABLESPACE
System Audit Settings/Audit Commands	AUDIT ANY
	AUDIT SYSTEM
System Audit Settings/Access Control	GRANT ANY PRIVILEGE/ANY OBJECT PRIVILEGE/ROLE
	GRANT DIRECTORY
	GRANT SEQUENCE
	GRANT TABLE
	GRANT TYPE

Table A–2 (Cont.) Audit Policy Settings Oracle Database Vault Adds to Oracle Database

Audit Setting Type	Audited Statements (BY ACCESS and on Success or Failure Unless Otherwise Noted)
User Audit Settings for DVSYS/DVF User Audit Settings for LBACSYS See Table 10–2, " Database Accounts Used by Oracle Database Vault" on page 10-9 for more information about these accounts. See also these sections for detailed information on the DVSYS and DVF schemas: ■ "DVSYS Schema" on page 10-1 ■ "DVF Schema" on page 10-2	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–2 (Cont.) Audit Policy Settings Oracle Database Vault Adds to Oracle Database

Audit Setting Type	Audited Statements (BY ACCESS and on Success or Failure Unless Otherwise Noted)
Object Audit Settings for DVF	AUDIT PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/TABLE
	COMMENT TABLE/VIEW
	DELETE TABLE/VIEW
	EXECUTE PACKAGE/PROCEDURE/FUNCTION (WHENEVER NOT SUCCESSFUL)
	GRANT PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/TABLE
	RENAME PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/VIEW/TABLE
	SELECT SEQUENCE/TABLE/VIEW (WHENEVER NOT SUCCESSFUL)
Object Audit Settings for DVSYS	AUDIT PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/TABLE
Object Audit Settings for LBACSYS	COMMENT TABLE/VIEW
	DELETE TABLE/VIEW
	EXECUTE PACKAGE/PROCEDURE/FUNCTION (WHENEVER NOT SUCCESSFUL)
	GRANT PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/TABLE
	INSERT TABLE/VIEW
	RENAME PACKAGE/PROCEDURE/FUNCTION/SEQUENCE/VIEW/TABLE
	SELECT SEQUENCE/TABLE/VIEW (WHENEVER NOT SUCCESSFUL)
	UPDATE TABLE/VIEW

B

Disabling and Enabling Oracle Database Vault

This appendix contains:

- When You Must Disable Oracle Database Vault
- Checking if Oracle Database Vault Is Enabled or Disabled
- 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

You may need to disable Oracle Database Vault to perform upgrade tasks or correct erroneous configurations. You can reenable Oracle Database Vault after you complete the corrective tasks.

> **Note:** Be aware that if you disable Oracle Database Vault, the privileges that were revoked from existing users and roles during installation remain in effect. See "Privileges That Are Revoked or Prevented from Existing Users and Roles" on page 2-3 for a listing of the revoked privileges.

The following situations require you to disable Oracle Database Vault:

- The Oracle Database Vault user accounts have been inadvertently locked or their passwords forgotten. (See the tip under "Oracle Database Vault Accounts" on page 10-9 for a guideline for avoiding this problem in the future.)
- 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 or features, such as Oracle Spatial Data Option, Oracle OLAP, 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 must archive the Oracle Database Vault audit trail.

Checking if Oracle Database Vault Is Enabled or Disabled

You can check if Oracle Database Vault has already been enabled or disabled by querying the `V$OPTION` table. Any user can query this table. If Oracle Database Vault is enabled, the query returns `TRUE`. Otherwise, it returns `FALSE`.

Remember that the `PARAMETER` column value is case sensitive. For example:

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

If Oracle Database Vault is enabled, the following output appears:

```
PARAMETER                      VALUE
------------------------------ -----------------------
Oracle Database Vault          TRUE
```

Step 1: Disable Oracle Database Vault

To disable Oracle Database Vault:

1. Stop the database, Database Control console process, and listener.
   - **UNIX:** Ensure that the environment variables, `ORACLE_HOME`, `ORACLE_SID`, and `PATH` are correctly set. Log in to SQL*Plus as user `SYS` with the `SYSOPER` privilege and shut down the database. Then from the command line, stop the Database Control console process and listener.

     For example:

     ```
     sqlplus sys as sysoper
     Enter password: password

     SQL> SHUTDOWN IMMEDIATE
     SQL> EXIT

     $ emctl stop dbconsole
     $ lsnrctl stop [listener_name]
     ```

     For Oracle RAC installations, shut down each database instance as follows:

     ```
     $ srvctl stop database -d db_name
     ```

   - **Windows:** Stop the database, Database Control console process, and listener from the Services tool in the Control Panel. The names of Oracle Database services begin with `Oracle`.
2. Disable the Oracle Database Vault option.
   - **UNIX:** Run the following commands:

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

     For Oracle RAC installations, run these commands on all nodes.

   - **Windows:** In the *ORACLE_HOME*`\bin` directory, rename the `oradv10.dll` file to another name, such as `oradv10.dll.dbl`.

3. Restart the database, Database Control console process, and listener.

   - **UNIX:** Log in to SQL*Plus as user `SYS` with the `SYSOPER` privilege and restart the database. Then from the command line, restart the Database Control console process and listener.

     For example:

     ```
     sqlplus sys as sysoper
     Enter password: password

     SQL> STARTUP
     SQL> EXIT

     $ emctl start dbconsole
     $ lsnrctl start [listener_name]
     ```

     For Oracle RAC installations, restart each database instance as follows:

     ```
     $ srvctl start database -d db_name
     ```

   - **Windows:** Restart the database, Database Control console process, and listener from the Services tool in the Control Panel. The names of Oracle Database services begin with `Oracle`.

4. If the reason you needed to disable Oracle Database Vault was because of forgotten passwords, then connect as `SYS` or `SYSTEM` and reset the password.

   For example:

   ```
   CONNECT SYSTEM
   Enter password: password

   ALTER USER lbrown_dvowner IDENTIFIED BY password;
   ```

5. At a command prompt, run Oracle Database Vault Configuration Assistant (DVCA) by using the `dvca -action disable` option.

   The syntax for s disable is as follows:

   ```
   dvca -action disable
     -oh Oracle_home_directory
     -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.
   - `-oh` is the path to the Oracle home directory. Enter the absolute path.

     For example:

     - **UNIX:** `-oh /u01/app/oracle/product/10.2.0/db_1`
     - **Windows:** `-oh c:\oracle\product\db_1`
   - `-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. Preferably, omit the password and then enter it interactively when prompted.
- `-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. Preferably, omit the password and then enter it interactively when prompted.
- `-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 /u01/app/oracle/product/10.2.0/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. You can perform the following types of activities:

- **Use the Oracle Database Vault PL/SQL packages and functions.** For example, to correct a login or CONNECT rule set error, use the `DBMS_MACADM` package or the Oracle Database Vault Administrator interface.
- **Use the SYSTEM or SYS accounts to perform tasks such as creating or changing passwords.** In addition to modifying regular database and administrative user accounts, you can modify passwords and the lock status of any of the Oracle Database Vault-specific accounts, such as users who have been granted the `DV_ADMIN` or `DV_ACCTMGR` roles. (See the tip under "Oracle Database Vault Accounts" on page 10-9 for a guideline for avoiding this problem in the future.)
- **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 [listener_name]
```

Step 3: Enable Oracle Database Vault

To enable Oracle Database Vault:

1. At a command prompt, use DVCA to re-enable Oracle Database Vault.

   For example:

   ```
   dvca -action enable
     -oh /u01/app/oracle/product/10.2.0/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 "Step 1: Disable Oracle Database Vault" on page B-2 for detailed information about the DVCA syntax.

2. Stop the database, Database Control console process, and listener.

   - **UNIX:** Ensure that the environment variables, `ORACLE_HOME`, `ORACLE_SID`, and `PATH` are correctly set. Log in to SQL*Plus as user `SYS` with the `SYSOPER` privilege and shut down the database. Then from the command line, stop the Database Control console process and listener.

     For example:

     ```
     sqlplus sys as sysoper
     Enter password: password

     SQL> SHUTDOWN IMMEDIATE
     SQL> EXIT

     $ emctl stop dbconsole
     $ lsnrctl stop [listener_name]
     ```

     For Oracle RAC installations, shut down each database instance as follows:

     ```
     $ srvctl stop database -d db_name
     ```

   - **Windows:** Stop the database, Database Control console process, and listener from the Services tool in the Control Panel. The names of Oracle Database services begin with `Oracle`.

3. Enable the Oracle Database Vault option as follows:

   - **UNIX:** Run the following commands. The `make` command enables both Oracle Database Vault (`dv_on`) and Oracle Label Security (`lbac_on`). You must enable Oracle Label Security before you can use Database Vault.

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

   - **Windows:** In the *ORACLE_HOME*`\bin` directory, rename the backed up copy of the `oradv10.dll` file (for example, `oradv10.dll.dbl`) to `oradv10.dll`. Ensure that the name of the Oracle Label Security executable is `oraLbac10.dll` (and not `oraLbac10.dll.dbl` or some other backup

name). You must enable Oracle Label Security before you can use Database Vault.

4. Restart the database, Database Control console process, and listener.

   - **UNIX:** Ensure that the environment variables, `ORACLE_HOME`, `ORACLE_SID`, and `PATH` are correctly set. Log in to SQL*Plus as user `SYS` with the `SYSOPER` privilege and restart the database. Then from the command line, restart the Database Control console process and listener.

     For example:

     ```
     sqlplus sys as sysoper
     Enter password: password

     SQL> STARTUP
     SQL> EXIT

     $ emctl start dbconsole
     $ lsnrctl start [listener_name]
     ```

     For Oracle RAC installations, restart each database instance as follows:

     ```
     $ srvctl start database -d db_name
     ```

   - **Windows:** Restart the database, Database Control Console process, and listener from the Services tool in the Control Panel. The names of Oracle Database services begin with `Oracle`.

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

C

Oracle Database Vault Security Guidelines

This appendix contains:

- Separation of Duty Guidelines
- Managing Oracle Database Administrative Accounts
- Accounts and Roles Trusted by Oracle Database Vault
- Accounts and Roles That Should be Limited to Trusted Individuals
- Guidelines for Using Oracle Database Vault in a Production Environment
- Secure Configuration Guidelines

Separation of Duty Guidelines

This section contains:

- How Oracle Database Vault Handles Separation of Duty
- Defining Separate Tasks in an Oracle Database Vault Environment
- Creating a Separation of Duty Matrix
- Identifying and Documenting the Tasks of Users Who Access the Database System

How Oracle Database Vault Handles Separation of Duty

Separation of duty means that you restrict each user's privileges *only* to the tasks he or she is responsible for, and *no more*. You should assign specific categories of privileges to specific users, rather than granting many privileges to one user. Simply put, separation of duty creates accountability for each task that your organization requires.

Separation of duty has taken on increased importance over the past 10 years. For many organizations, separation of duty is a new concept that continues to evolve. Database consolidation, regulatory compliance, and outsourcing are just a few of the drivers for increased separation of duty. Oracle Database Vault separation of duty strengthens security by separating security-related administration from day-to-day DBA operations. You can tailor your Database Vault separation of duty implementation to easily adapt to current and future business requirements. Small organizations, in particular, need flexibility as they attempt to increase their security profile with limited resources.

Defining Separate Tasks in an Oracle Database Vault Environment

Oracle Database Vault defines the following main responsibilities:

- **Account management.** Account management entails creating, modifying, and dropping user accounts. The `DV_ACCTMGR` role provides these privileges.
- **Security administration.** Security administration covers basic security tasks such as creating realms and command rules, setting security policies for database users' access, and authorizing database users for jobs they are allowed to perform. Security administrators also run security audit reports. The `DV_OWNER` and `DV_ADMIN` roles provide these privileges. (For an in-depth look at how the Oracle Database Vault roles provide for separation of duty, see "Oracle Database Vault Roles" on page 10-2.)

  Optionally, you can consolidate the account management and security administrative responsibilities.
- **Resource management.** Resource management refers to managing the database system but not accessing business data. It includes the following operations:
  - Backup operations require a predefined time to perform the backup using predefined tools.
  - Tuning and monitoring operations require ongoing performance monitoring and analysis.
  - Patching operations require temporary access only during the time the patching takes place

  For resource management, you should create a named account and a backup account for each of these tasks. Add these accounts as owners of the Data Dictionary realm. Use these accounts as the primary resource managers in the database.

You should have separate accounts for database account management, database security administration, and additional named accounts for backup operations. Auditors check for separate database accounts for different responsibilities and being able to track the actions of each account. Less important is the number of users assigned to specific tasks. Remember that Oracle Database Vault audit events are protected and that the Database Vault reports show all attempted violations.

Creating a Separation of Duty Matrix

Before separation of duty can be successful, you must understand who performs basic administration tasks in your environment and what these administration tasks are. Even if a single database administrator is responsible for managing both new database account provisioning and application patching, it is important to document and plan for each of these tasks.Using separate administration accounts for these types of tasks provides increased accountability and reduces associated risks. In midsize to large organizations, database administrators typically must perform common administration tasks but they do not need access to business data managed by the application. Creating a matrix for your separation of duty can help you plan your Database Vault deployment. As needed, you can include additional tasks and associated users to this list. This information should become part of the overall enterprise security documentation for your organization.

Table C–1 shows an example of a separation of duty matrix.

Table C–1 Example Separation of Duty Matrix

User, Process or Application	Account Creation	Database Administration					Security Administrator
		SYSDBA	Backup	Tuning	Patching	Monitoring	
JSMITH	X						
SHARDY							X
PKESTNER			X				
RTYLER					X		
SANDERSON				X		X	
SYSTEM					EBS patching		
RMAN		X	X				
...							

In some cases, system management tasks may require temporary access to data through specific tools and programs. When this happens, build provisions for this temporary or emergency access into the Oracle Database Vault rules and rule sets.

Identifying and Documenting the Tasks of Users Who Access the Database System

You should document the following areas of the tasks your organization needs:

- The responsibilities of each administrative user
- The kind of access users need. For example, application owners should have data access and developers need access to development instances only.
- Who must manage the system without accessing business data (for example, users who perform backup, patching, tuning, and monitoring operations).
- The duties of each category of tasks (for example, the files that must be backed up, the applications that require patching, what exactly is monitored). Include the alternate user accounts for each of these tasks.
- The databases and applications that must be protected. This includes Oracle applications, partner applications, and custom applications.
- Who must be authorized to access business data, including the following:
  - Application owners through middle tier processes
  - Business users through an application interface
- Emergency "what if" scenarios, such as how to handle a security breach
- Reporting in a production environment, which should include the following:
  - Who runs the reports
  - Which reports must be run
  - The frequency with which each report is run
  - The users who must receive a copy of each report
- In addition to a separation of duty matrix, the creation of the following matrices:
  - An Oracle Database Vault-specific matrix, which can cover the names and tasks of users who have been granted Database Vault roles

– An application protection matrix, which can cover the applications to be protected and the types of protections you have put in place.

Table C–2 shows an example of protections Oracle created for PeopleSoft Applications. You can download the scripts to create these security policies from the following URL:

http://www.oracle.com/technology/software/products/database_vault/index.html

Table C–2 Example Application Protection Matrix

	Authorized with Rule Set			
Protection Type	**SYSADM**	**PSFTDBA**	**SYSTEM**	**DBA**
PeopleSoft Realm	Owner	Owner	No Access	No Access
SELECT Command Rule	Not Restricted	Limit PSFTDB Rule Set	No Access	No Access
CONNECT Command Rule	PeopleSoftAccess Rule Set	Not Restricted	Not Restricted	Not Restricted
DROP TABLESPACE Command Rule	Disabled Rule Set	Disabled Rule Set	Disabled Rule Set	Disabled Rule Set

Managing Oracle Database Administrative Accounts

This section contains:

- Using the SYSTEM User Account for General Administrative Uses
- Using the SYSTEM Schema for Application Tables
- Limiting the SYSDBA Privilege
- Managing Root and Operating System Access

Using the SYSTEM User Account for General Administrative Uses

If you use the SYSTEM account for general database administrative purposes, create named database administrative accounts for your database administrators. Doing so increases accountability for administrative actions in the database.

Using the SYSTEM Schema for Application Tables

If your site holds application tables in the SYSTEM schema, then you should add the SYSTEM account to your realm authorizations for these tables so that these applications can continue to work normally. You can place restrictions on the SYSTEM account to increase or fine-tune security for these applications. For example, you can create a Database Vault rule set to restrict the SYSTEM user's access to specific IP addresses.

Limiting the SYSDBA Privilege

Limit the SYSDBA privilege only to users who must connect using this privilege when absolutely necessary and for applications that still require SYSDBA access, such as Oracle Recovery Manager (RMAN) and mandatory patching processes. For all other cases, create named database accounts to perform daily database administration.

Managing Root and Operating System Access

Oracle Database Vault prevents highly privileged database users from accessing sensitive data. In addition, if you are using Oracle Database itself, you can use transparent data encryption to prevent the most highly privileged operating system users from accessing sensitive data. Transparent data encryption enables you to hide individual table columns. (See *Oracle Database Advanced Security Administrator's Guide* for more information about transparent data encryption.) As a best practice, always carefully review and restrict direct access to the operating system.

You should have personalized accounts access the operating system. These personalized accounts should, in the Linux or UNIX environments, login using `sudo` to the `oracle` software owner when needed. With `sudo`, you can control which specific command each personalized user can execute. Be sure to prevent the use of the `make`, `relink`, `gdb`, or other commands that could potentially harm the database, for these users. However, if an administrative user must install a patch or perform some other emergency operation, you can enable the `make` and `relink` commands for a limited time, and audit their actions during this period.

Accounts and Roles Trusted by Oracle Database Vault

Oracle Database Vault restricts access to application data from many privileged users and roles in the database. However, in some cases, Oracle Database Vaults trusts certain roles and privileges.

Table C–3 lists the trusted roles and privileges that are created when you install Oracle Database Vault.

Table C–3 Trusted Oracle Database Vault Roles and Privileges

Role or Privilege	Status	Description
`DV_ACCTMGR` role	Open	Role created during installation and used for creating new database accounts
`DV_OWNER` role	Open	Role created during installation and used for managing realms, factors and command rules. This user 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 C-6 for guidelines on managing `SYSDBA`.
`SYSOPER` privilege	Enabled	Privilege created during Oracle Database installation. Database startup and shutdown. Granted to `SYS` only by default. See "Managing SYSOPER Access" on page C-6 for guidelines on managing `SYSOPER`.

Accounts and Roles That Should be Limited to Trusted Individuals

Several accounts and roles have very powerful privileges in a default Oracle Database installation. You should limit these accounts and roles only to trusted individuals.

- 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

The `SYSDBA` privilege is a trusted privilege in Oracle Database Vault. Grant this privilege to trusted users only.

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` privilege has limited privileges within the database, but individuals with this role can start and shut down the Oracle database. Only grant the `SYSOPER` privilege to trusted individuals.

Guidelines for Using Oracle Database Vault in a Production Environment

Follow these guidelines when running Oracle Database Vault in a production environment:

- Run a full test of your applications to ensure that the Database Vault policies you have created are working as expected
- Monitor the performance of your applications, and if necessary, tune your rule expressions
- Assign responsibilities to the appropriate production support and security groups, as follows:
  - Assign security responsibilities to the database security administrator.

  - Assign account management to the database account manager.
  - Assign resource management tasks to database administrators.
- Back up your Database Vault API scripts to a secure server.

Secure Configuration Guidelines

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:

- 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 to SQL*Plus as `SYSTEM` and issue the following query.

  ```
  SELECT * FROM DBA_TAB_PRIVS WHERE TABLE_NAME package_name;
  ```

  package_name is the name of the package you are looking for.

  To find the users, packages, procedures, and functions that are dependent on the package, issue this query:

  ```
  SELECT OWNER, NAME, TYPE  FROM ALL_DEPENDENCIES
  WHERE REFERENCED_NAME = package_name;
  ```

  Note that these two queries do not identify references to packages made through dynamic SQL.

See Also:

- "Privileges That Are Revoked or Prevented from Existing Users and Roles" on page 2-3
- Table 10–1, " Privileges of Oracle Database Vault Roles" on page 10-3

Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages

The UTL_FILE package is owned by SYS and granted to PUBLIC. However, a user must have access to the directory object to manipulate the files in that operating system directory. You can configure the UTL_FILE package securely; see *Oracle Database PL/SQL Packages and Types Reference* for more information.

The DBMS_FILE_TRANSFER package is owned by SYS and granted to the EXECUTE_CATALOG_ROLE. Users with EXECUTE access on this package can move files from one location to another on the same file system. They also can move files between database instances, including databases on remote systems.

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 C–1 and Example C–2 show how you can use the Oracle Database Vault DVSYS.DBMS_MACADM package to create command rules that limit and enable access to the CREATE DATABASE LINK statement, which 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 C–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 valid user must use the CREATE DATABASE LINK statement, the Oracle Database Vault owner can reenable it from Oracle Database Vault Administrator or issue the following commands in SQL*Plus.

Example C–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 C–3 shows command rules that disable and enable access to CREATE DIRECTORY.

Example C–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. The ALTER SYSTEM command rule prevents it from being enabled. If you need to use the recycle bin, log in to SQL*Plus, disable the command rule ALTER SYSTEM, and then enable the recycle bin.

Remember, however, that when the recycle bin feature is enabled, any realm-protected object that is dropped will go into the recycle bin. Once there, it is no longer protected by the realm. In SQL*Plus, you can check the contents of the recycle bin as follows:

```
SELECT * FROM RECYCLEBIN;
SELECT * FROM USER_RECYCLEBIN;
```

To purge the contents of the recycle bin, use the PURGE SQL statement:

```
PURGE RECYCLEBIN;
PURGE USER_RECYCLEBIN;
```

To disable the ALTER SYSTEM command rule and enable the recycle bin:

1. As the DVOWNER or DV_ADMIN account, disable the ALTER SYSTEM command rule.

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'ALTER SYSTEM',
  rule_set_name   => 'Allow System Parameters',
  object_owner    => '%',
  object_name     => '%',
  enabled         => 'N');
END;
/
```

2. Connect as SYS/AS SYSDBA and then enable the RECYCLE BIN.

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

ALTER SYSTEM SET RECYCLEBIN=ON SCOPE=SPFILE;
```

3. Connect as DVOWNER or DV_ADMIN and then re-enable the ALTER SYSTEM command rule.

```
BEGIN
 DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'ALTER SYSTEM',
  rule_set_name   => 'Allow System Parameters',
  object_owner    => '%',
  object_name     => '%',
  enabled         => 'Y');
END;
/
```

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 the EXECUTE privilege 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. You should grant the ALTER SESSION privilege only to trusted users. (For example, the ALTER SESSION statement can enable tracing.)

Example C–4 shows how you can create this type of 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 C–4 Adding Rules to the Existing ALTER SYSTEM Command Rule

```
CONNECT amalcolm_dvacctmgr
Enter password: password

BEGIN
 DBMS_MACADM.CREATE_RULE('NO_SYSTEM_DUMP',
 '(INSTR(UPPER(DVSYS.DV_SQL_TEXT),''DUMP'') = 0)');
 END;
/
EXEC DBMS_MACADM.ADD_RULE_TO_RULE_SET
  ('Allow System Parameters','NO_SYSTEM_DUMP');

EXEC DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE
   ('ALTER SYSTEM', 'Allow System Parameters', '%', '%', 'Y');
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, EXP_FULL_DATABASE, and IMP_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: Identify the Java Stored Procedures Created with Definer's Rights
- Step 2: Find the Java Stored Procedures That Access Realm-Protected Objects
- Step 3: Create a Package to Wrap Procedures Accessing Realm-Protected Objects
- Step 4: Identify the Java Stored Procedures Created with Invoker's Rights
- Step 5: Block the Execution of Java Stored Procedures
- Step 6: Verify Oracle Database Vault Protection for Java Stored Procedures
- Step 7: Secure the Invoker's Rights for New Java Stored Procedures

Step 1: Identify 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 C–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 C–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: Find the 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 "DBA_DV_REALM_OBJECT View" on page 10-22.

Step 3: Create 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 C–6 shows the PL/SQL package `mypackage` being created to wrap the Java class `emp_count`.

Example C–6 Creating a PL/SQL Wrapper

```
CREATE OR REPLACE PACKAGE SCOTT.MYPACKAGE AS
 tmp varchar2(200) := 'TEST';  -- dummy variable
 FUNCTION empcount RETURN VARCHAR2;
END;
/

CREATE OR REPLACE PACKAGE BODY SCOTT.MYPACKAGE AS
   FUNCTION empcount RETURN VARCHAR2 AS LANGUAGE JAVA
   NAME 'emp_count.count() return java.lang.String';
END;
/
```

Step 4: Identify 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 C–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 C–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: Block the 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: Create a Package to Wrap Procedures Accessing Realm-Protected Objects.

Step 6: Verify Oracle Database Vault Protection for Java Stored Procedures

Verify that Oracle Database Vault is protecting your Java stored procedures. Example C–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 C–8 Testing Oracle Database Vault Protection for Java Stored Procedures

```
SQL> CONNECT u1
Enter password: password

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;

ERROR at line 1:
ORA-01031: insufficient privileges

--Now show protecting access through Java

SQL> SELECT SCOTT.MYPACKAGE.EMPCOUNT FROM DUAL;

ERROR at line 1:
ORA-01031: insufficient privileges
ORA-06512: at "SCOTT.MYPACKAGE", line 2
```

Step 7: Secure the 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`, `EXP_FULL_DATABASE`, and `IMP_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 External C Callouts
- 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 C–9. This query spools the results to the file `external_wrap.lst`.

Example C–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 "DBA_DV_REALM_OBJECT View" on page 10-22.

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 C–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 C–11. This query spool the results to the file `external_standalone.lst`.

Example C–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 External C Callouts

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 C–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 C–12 Testing Oracle Database Security for an External C Callout

```
SQL> CONNECT u1
Enter password: password

SQL> SELECT * FROM SESSION_PRIVS;

PRIVILEGE
----------------------------------------
CREATE SESSION
SELECT ANY TABLE
CREATE PROCEDURE
EXECUTE ANY PROCEDURE

SQL> SELECT COUNT(*) FROM SCOTT.EMP;

ERROR at line 1:
ORA-01031: insufficient privileges

SQL> SELECT TEST FROM DUAL;
```

```
TEST
--------------------------------------------------------------------------------
14

SQL> 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.

D

Troubleshooting Oracle Database Vault

This appendix contains:

- 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, and 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.

The `USER_DUMP_DEST` initialization parameter specifies the current location of the trace files. You can find the value of this parameter by issuing `SHOW PARAMETER USER_DUMP_DEST` in SQL*Plus.

To enable tracing, log in to SQL*Plus as a user who has the `ALTER SESSION` privilege and then issue the following statement:

```
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:

```
ALTER SESSION SET EVENTS '47998 trace name context off'
```

For more information about how to manage trace files, see *Oracle Database Administrator's 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.
- If Database Vault Administrator does not display the default realms, command rules, rule sets, or factors, then Oracle Database and your operating system have conflicting NLS settings.

  The value of the initialization parameter `NLS_LANGUAGE` in Oracle Database must match up correctly with the locale and NLS settings at the operating system level (either `NLS_LANG` or `LANG` environment variables). For example, if the operating system locale (the variable `$LANG`) setting is `en_US.UTF-8`, then the corresponding `NLS_LANG` environment variable should be set to `AMERICAN_AMERICA.AL32UTF8` and the database `NLS_LANGUAGE` parameter value will be `AMERICAN`. The database `NLS_LANGUAGE` parameter is derived from the operating system `NLS_LANG` environment variable.

  For more information about checking and configuring locale and NLS settings, see the appendix that covers globalization support in the *Oracle Database Installation Guide* for your platform.
- 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 16-4 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 16-3
- "Factor Configuration Issues Report" on page 16-3
- "Factor Without Identities Report" on page 16-3
- "Identity Configuration Issues Report" on page 16-3
- "Realm Authorization Configuration Issues Report" on page 16-3
- "Rule Set Configuration Issues Report" on page 16-4
- "Secure Application Configuration Issues Report" on page 16-4

To run these reports, see "How to Run Oracle Database Vault Reports" on page 16-2.

Index

Symbols

% wildcard, 16-2

A

access control policy
  configuring with tools and components
    Oracle Label Security PL/SQL APIs, 1-3
    Oracle Policy Manager, 1-3
  reports
    Core Database Vault Audit Report, 16-5
access control run-time PL/SQL procedures and functions, 14-1
Access to Sensitive Objects Report, 16-8
accounts
  See database accounts
Accounts With DBA Roles Report, 16-10
Accounts with SYSDBA/SYSOPER Privilege Report, 16-8
ad hoc tools
  preventing use of, 7-17
administrators
  restricting different types, 7-22
alerts
  Enterprise Manager Grid Control, 9-4
ALTER DATABASE statement
  monitoring, 15-2
ALTER ROLE statement
  monitoring, 15-3
ALTER SESSION privilege
  enabling trace files, D-1
  reports, ALTER SYSTEM or ALTER SESSION Report, 16-10
ALTER SESSION statement
  guidelines on managing privileges, C-10
ALTER SYSTEM or ALTER SESSION Report, 16-10
ALTER SYSTEM privilege
  reports, ALTER SYSTEM or ALTER SESSION Report, 16-10
ALTER SYSTEM statement
  controlling with command rules, 6-1
  guidelines on managing privileges, C-10
ALTER TABLE statement
  monitoring, 15-2
ALTER USER statement
  monitoring, 15-3
ANY privileges, 10-7
ANY System Privileges for Database Accounts Report, 16-7
APIs
  See DVSYS.DBMS_MACADM package, DVSYS.DBMS_MACSEC_ROLES package, DVSYS.DBMS_MACUTL package
AUD$ table
  See SYS.AUD$ table
audit policy change
  monitoring, 15-3
AUDIT privilege, 16-11
AUDIT Privileges Report, 16-11
AUDIT_SYS_OPERATIONS initialization parameter, 2-1
AUDIT_TRAIL initialization parameter
  effect on Core Database Audit Report, 16-12
AUDIT_TRAIL$ system table
  affected by AUDIT_TRAIL initialization parameter, A-2
  archiving, A-4
  format, A-2
  purging, A-4
auditing
  archiving Database Vault audit trail, A-4
  Core Database Audit Report, 16-12
  DVSYS.DBMS_MACUTL fields, 13-1
  factors
    options, 7-9
  intruders
    using factors, 7-10
    using rule sets, 5-4
  Oracle Database audit settings, A-6 to A-9
  purging Database Vault audit trail, A-4
  realms
    DVSYS.DBMS_MACUTL fields, 13-3
    options, 4-3
  reports, 16-4 to 16-5
  rule sets
    DVSYS.DBMS_MACUTL fields, 13-3
    options, 5-3
  secure application roles
    audit records, 8-9
  troubleshooting, D-1
  views used to audit events, 10-13
auditing policies

about, A-1
audit events
about, A-1
custom events
audit trail, A-2
events that are tracked, A-1
monitoring changes to, 15-3
authentication
Authentication_Method default factor, 7-2
command rules, 6-2
method, finding with DVF.F$AUTHENTICATION_METHOD, 14-7
realm procedures, 11-2
authorizations, realms, 4-5

B

BECOME USER Report, 16-10
BECOME USER system privilege
about, 16-10

C

catalog-based roles, 16-11
child factors
See factors
clients
finding IP address with DVF.F$CLIENT_IP, 14-8
code groups
retrieving value with DVSYS.DBMS_MACUTL functions, 13-5
Command Rule Audit Report, 16-4
Command Rule Configuration Issues Report, 16-3
command rules
about, 6-1
creating, 6-4
data dictionary view, 6-10
default command rules, 6-2
default command rules not showing in Database Vault Administrator, D-2
deleting, 6-5
diagnosing behavior, D-1
editing, 6-4
functions
DVSYS.DBMS_MACUTL (utility), 13-1 to 13-6
guidelines, 6-8
how command rules work, 6-5
objects
name, 6-5
owner, 6-5
performance effect, 6-9
procedures
DVSYS.DBMS_MACADM (configuration), 11-22
process flow, 6-5
propagating policies to other databases, 9-2
reports, 6-9
rule sets
selecting, 6-5
used with, 6-1
troubleshooting
general diagnostic advice, D-1
with auditing report, 16-4
tutorial, 6-6
views, 6-10, 10-14
See also rule sets
compliance
Oracle Database Vault addressing, 1-4
computer name
finding with DVF.F$MACHINE, 14-12
Machine default factor, 7-4
configuration
changes, monitoring, 15-3
See also DVSYS.DBMS_MACADM package
CONNECT events, controlling with command rules, 6-2
core database
troubleshooting with Core Database Vault Audit Report, 16-5
Core Database Audit Report, 16-12
Core Database Vault Audit Trail Report, 16-5
CPU_PER_SESSION resource profile, 16-12
CREATE ANY JOB privilege, C-10
CREATE ANY JOB statement
guidelines on managing privileges, C-10
CREATE EXTERNAL JOB privilege, C-10
CREATE JOB privilege, C-10
CREATE JOB statement
guidelines on managing privileges, C-10
CREATE ROLE statement
monitoring, 15-3
CREATE TABLE statement
monitoring, 15-2
CREATE USER statement
monitoring, 15-3

D

data definition language (DDL)
statement
controlling with command rules, 6-1
data dictionary
adding DV_ACCTMGR role to realm, 3-6
data manipulation language (DML)
statement
checking with DVSYS.DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED function, 13-5
controlling with command rules, 6-1
data Oracle Database Vault recognizes
See factors
Database Account Default Password Report, 16-12
Database Account Status Report, 16-12
database accounts
counting privileges of, 16-9
DBSNMP
realm for, 4-2
DVSYS, 10-9
LBACSYS, 10-10
monitoring, 15-3
reports
Accounts With DBA Roles Report, 16-10

ALTER SYSTEM or ALTER SESSION Report, 16-10
ANY System Privileges for Database Accounts Report, 16-7
AUDIT Privileges Report, 16-11
BECOME USER Report, 16-10
Database Account Default Password Report, 16-12
Database Account Status Report, 16-12
Database Accounts With Catalog Roles Report, 16-11
Direct and Indirect System Privileges By Database Account Report, 16-7
Direct Object Privileges Report, 16-6
Direct System Privileges By Database Account Report, 16-7
Hierarchical System Privileges by Database Account Report, 16-7
Object Access By PUBLIC Report, 16-6
Object Access Not By PUBLIC Report, 16-6
OS Security Vulnerability Privileges, 16-11
Password History Access Report, 16-11
Privileges Distribution By Grantee Report, 16-9
Privileges Distribution By Grantee, Owner Report, 16-9
Privileges Distribution By Grantee, Owner, Privilege Report, 16-9
Roles/Accounts That Have a Given Role Report, 16-11
Security Policy Exemption Report, 16-10
WITH ADMIN Privilege Grants Report, 16-10
WITH GRANT Privileges Report, 16-11
solution for lockouts, B-1
suggested, 10-10
SYSMAN
realm for, 4-2
Database Accounts With Catalog Roles Report, 16-11
database configuration
monitoring changes, 15-2
database definition language (DDL)
statements
controlling with command rules, 6-1
database domains, Database_Domain default factor, 7-2
database objects
Oracle Database Vault, 10-1 to 10-26
reports
Object Dependencies Report, 16-6
See also objects
database options, installing, B-1
database roles
about, 10-2
counting privileges of, 16-9
default Oracle Database Vault, 10-2
DV_ACCTMGR
about, 10-6
adding to Data Dictionary realm, 3-6
DV_ADMIN, 10-6
DV_OWNER, 10-4
DV_PUBLIC, 10-7
DV_REALM_OWNER, 10-5
DV_REALM_RESOURCE, 10-5
DV_SECANALYST, 10-8
enabled, determining with DVSYS.ROLE_IS_ENABLED, 14-4
monitoring, 15-3
Oracle Database Vault, default, 10-2
reports
Accounts With DBA Roles Report, 16-10
ALTER SYSTEM or ALTER SESSION Report, 16-10
AUDIT Privileges Report, 16-11
BECOME USER Report, 16-10
Database Accounts With Catalog Roles Report, 16-11
OS Security Vulnerability Privileges, 16-11
Privileges Distribution By Grantee Report, 16-9
Roles/Accounts That Have a Given Role Report, 16-11
Security Policy Exemption Report, 16-10
WITH ADMIN Privilege Grants Report, 16-10
separation of duty enforcement, 2-3
database schemas
grouped
See realms
database sessions, 7-6
controlling with Allow Sessions default rule set, 5-2
factor evaluation, 7-15
session user name, Proxy_User default factor, 7-4
Database Vault
See Oracle Database Vault
Database Vault Administrator
setting URL in Oracle Enterprise Manager, 9-1
databases
dbconsole
checking process, 3-4
starting process, 3-4
defined with factors, 7-1
domain, Domain default factor, 7-3
event monitoring, D-1
host names, Database_Hostname default factor, 7-3
instance, retrieving information with functions, 11-26
instances
Database_Instance default factor, 7-3
names, finding with DVF.F$DATABASE_INSTANCE, 14-9
number, finding with DVSYS.DV_INSTANCE_NUM, 14-15
IP addresses
Database_IP default factor, 7-3
retrieving with DVF.F$DATABASE_IP, 14-9
listener, starting, B-4
log file location, 3-5
monitoring events, D-1

names
Database_Name default factor, 7-3
retrieving with DVF.F$DATABASE_NAME, 14-9
retrieving with DVSYS.DV_DATABASE_NAME, 14-16
parameters
Security Related Database Parameters Report, 16-11
roles that do not exist, 16-4
schema creation, finding with DVF.F$IDENTIFICATION_TYPE, 14-11
schema creation, Identification_Type default factor, 7-3
structural changes, monitoring, 15-2
user name, Session_User default factor, 7-4
DBA_DV_CODE view, 10-12
DBA_DV_COMMAND_RULE view, 6-10, 10-14
DBA_DV_FACTOR view, 10-15
DBA_DV_FACTOR_LINK view, 10-16
DBA_DV_FACTOR_TYPE view, 10-17
DBA_DV_IDENTITY view, 10-17
DBA_DV_IDENTITY_MAP view, 10-18
DBA_DV_MAC_POLICY view, 10-19
DBA_DV_MAC_POLICY_FACTOR view, 10-19
DBA_DV_POLICY_LABEL view, 10-20
DBA_DV_PUB_PRIVS view, 10-20
DBA_DV_REALM view, 10-21
DBA_DV_REALM_AUTH view, 10-22
DBA_DV_REALM_OBJECT view, 10-22
DBA_DV_ROLE view, 10-23
DBA_DV_RULE view, 10-23
DBA_DV_RULE_SET view, 10-24
DBA_DV_RULE_SET_RULE view, 10-25
DBA_DV_USER_PRIVS view, 10-25
DBA_DV_USER_PRIVS_ALL view, 10-26
DBA_REGISTRY view, not applying to Database Vault components, 10-11
dbconsole process
checking status, 3-4
starting, 3-4
DBMS_FILE_TRANSFER package, guidelines on managing, C-8
DBMS_RLS PL/SQL package
Oracle Database Vault impact on, 2-5
DBSNMP user account
realm for, 4-2
DELETE_CATALOG_ROLE role, 16-11
Denial of Service (DoS) attacks
reports
System Resource Limits Report, 16-12
Tablespace Quotas Report, 16-14
Direct and Indirect System Privileges By Database Account Report, 16-7
Direct Object Privileges Report, 16-6
direct system privileges, 16-7
Direct System Privileges By Database Account Report, 16-7
disabling system features with Disabled default rule set, 5-2
domains
defined with factors, 7-1
finding database domain with DVF.F$DATABASE_DOMAIN, 14-8
finding with DVF.F$DOMAIN, 14-10
DROP ROLE statement
monitoring, 15-3
DROP TABLE statement
monitoring, 15-2
DROP USER statement
monitoring, 15-3
dual key connection, dual key security
See two-person integrity (TPI)
DV_ACCTMGR role
about, 10-6
adding to Data Dictionary realm, 3-6
DV_ADMIN role, 10-6
DV_OWNER role, 10-4
DV_PUBLIC role, 10-7
DV_REALM_OWNER role, 10-5
DV_REALM_RESOURCE role, 10-5
DV_SECANALYST role, 10-8
DVA
See Oracle Database Vault Administrator
DVCA
See Oracle Database Vault Configuration Assistant
DVF account
auditing policy, A-8
database accounts
DVF, 10-9
DVF schema, 14-5
about, 10-2
auditing policy, A-9
DVSYS account, 10-9
auditing policy, A-8
DVSYS schema
about, 10-1
auditing policy, A-9
command rules, 6-5
DV_OWNER role, 10-4
factor validation methods, 7-8
DVSYS.DBMS_MACADM package
about, 11-1
command rule procedures, listed, 11-22
factor procedures, listed, 11-26
Oracle Label Security policy procedures, listed, 11-48
realm procedures, listed, 11-1
rule set procedures, listed, 11-12
secure application role procedures, listed, 11-45
DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM procedure, 11-2
DVSYS.DBMS_MACADM.ADD_FACTOR_LINK procedure, 11-27
DVSYS.DBMS_MACADM.ADD_OBJECT_TO_REALM procedure, 11-4
DVSYS.DBMS_MACADM.ADD_POLICY_FACTOR procedure, 11-28
DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET procedure, 11-12

DVSYS.DBMS_MACADM.CHANGE_IDENTITY_FACTOR procedure, 11-28
DVSYS.DBMS_MACADM.CHANGE_IDENTITY_VALUE procedure, 11-29
DVSYS.DBMS_MACADM.CREATE_COMMAND_RULE procedure, 11-23
DVSYS.DBMS_MACADM.CREATE_DOMAIN_IDENTITY procedure, 11-30
DVSYS.DBMS_MACADM.CREATE_FACTOR procedure, 11-30
DVSYS.DBMS_MACADM.CREATE_FACTOR_TYPE procedure, 11-33
DVSYS.DBMS_MACADM.CREATE_IDENTITY procedure, 11-34
DVSYS.DBMS_MACADM.CREATE_IDENTITY_MAP procedure, 11-34
DVSYS.DBMS_MACADM.CREATE_MAC_POLICY procedure, 11-49
DVSYS.DBMS_MACADM.CREATE_POLICY_LABEL procedure, 11-50
DVSYS.DBMS_MACADM.CREATE_REALM procedure, 11-5
DVSYS.DBMS_MACADM.CREATE_ROLE procedure, 11-46
DVSYS.DBMS_MACADM.CREATE_RULE procedure, 11-14
DVSYS.DBMS_MACADM.CREATE_RULE_SET procedure, 11-14
DVSYS.DBMS_MACADM.DELETE_AUTH_FROM_REALM procedure, 11-6
DVSYS.DBMS_MACADM.DELETE_COMMAND_RULE procedure, 11-24
DVSYS.DBMS_MACADM.DELETE_FACTOR procedure, 11-35
DVSYS.DBMS_MACADM.DELETE_FACTOR_LINK procedure, 11-36
DVSYS.DBMS_MACADM.DELETE_FACTOR_TYPE procedure, 11-36
DVSYS.DBMS_MACADM.DELETE_IDENTITY procedure, 11-37
DVSYS.DBMS_MACADM.DELETE_IDENTITY_MAP procedure, 11-37
DVSYS.DBMS_MACADM.DELETE_MAC_POLICY_CASCADE procedure, 11-51
DVSYS.DBMS_MACADM.DELETE_OBJECT_FROM_REALM procedure, 11-7
DVSYS.DBMS_MACADM.DELETE_POLICY_FACTOR procedure, 11-52
DVSYS.DBMS_MACADM.DELETE_POLICY_LABEL procedure, 11-52
DVSYS.DBMS_MACADM.DELETE_REALM procedure, 11-8
DVSYS.DBMS_MACADM.DELETE_REALM_CASCADE procedure, 11-8
DVSYS.DBMS_MACADM.DELETE_ROLE procedure, 11-46
DVSYS.DBMS_MACADM.DELETE_RULE procedure, 11-16
DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET procedure, 11-17
DVSYS.DBMS_MACADM.DELETE_RULE_SET procedure, 11-18
DVSYS.DBMS_MACADM.DROP_DOMAIN_IDENTITY procedure, 11-38
DVSYS.DBMS_MACADM.GET_INSTANCE_INFO function, 11-39
DVSYS.DBMS_MACADM.GET_SESSION_INFO function, 11-40
DVSYS.DBMS_MACADM.RENAME_FACTOR procedure, 11-40
DVSYS.DBMS_MACADM.RENAME_FACTOR_TYPE procedure, 11-41
DVSYS.DBMS_MACADM.RENAME_REALM procedure, 11-9
DVSYS.DBMS_MACADM.RENAME_ROLE procedure, 11-47
DVSYS.DBMS_MACADM.RENAME_RULE procedure, 11-18
DVSYS.DBMS_MACADM.RENAME_RULE_SET procedure, 11-19
DVSYS.DBMS_MACADM.SYNC_RULES procedure, 11-19
DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE procedure, 11-24
DVSYS.DBMS_MACADM.UPDATE_FACTOR procedure, 11-41
DVSYS.DBMS_MACADM.UPDATE_FACTOR_TYPE procedure, 11-44
DVSYS.DBMS_MACADM.UPDATE_IDENTITY procedure, 11-44
DVSYS.DBMS_MACADM.UPDATE_MAC_POLICY procedure, 11-53
DVSYS.DBMS_MACADM.UPDATE_REALM procedure, 11-9
DVSYS.DBMS_MACADM.UPDATE_REALM_AUTH procedure, 11-10
DVSYS.DBMS_MACADM.UPDATE_ROLE procedure, 11-47
DVSYS.DBMS_MACADM.UPDATE_RULE procedure, 11-19
DVSYS.DBMS_MACADM.UPDATE_RULE_SET procedure, 11-20
DVSYS.DBMS_MACSEC_ROLES package
  about, 12-1
  functions, listed, 12-1
DVSYS.DBMS_MACSEC_ROLES.CAN_SET_ROLE function, 12-1
DVSYS.DBMS_MACSEC_ROLES.SET_ROLE procedure, 12-2
DVSYS.DBMS_MACUTL package
  about, 13-1
  constants (fields)
    examples, 13-4
    listed, 13-1
  procedures and functions, listed, 13-5
DVSYS.DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED procedure, 13-6
DVSYS.DBMS_MACUTL.GET_CODE_VALUE function, 13-7
DVSYS.DBMS_MACUTL.GET_DAY function, 13-9

DVSYS.DBMS_MACUTL.GET_HOUR
function, 13-9
DVSYS.DBMS_MACUTL.GET_MINUTE
function, 13-8
DVSYS.DBMS_MACUTL.GET_MONTH
function, 13-10
DVSYS.DBMS_MACUTL.GET_SECOND
function, 13-8
DVSYS.DBMS_MACUTL.GET_YEAR
function, 13-11
DVSYS.DBMS_MACUTL.IS_ALPHA function, 13-11
DVSYS.DBMS_MACUTL.IS_DIGIT function, 13-12
DVSYS.DBMS_MACUTL.IS_DVSYS_OWNER
function, 13-12
DVSYS.DBMS_MACUTL.IS_OLS_INSTALLED
function, 13-13
DVSYS.DBMS_MACUTL.IS_OLS_INSTALLED_
VARCHAR function, 13-13
DVSYS.DBMS_MACUTL.USER_HAS_OBJECT_
PRIVILEGE function, 13-14
DVSYS.DBMS_MACUTL.USER_HAS_ROLE
function, 13-15
DVSYS.DBMS_MACUTL.USER_HAS_ROLE_
VARCHAR function, 13-16
DVSYS.DBMS_MACUTL.USER_HAS_SYSTEM_
PRIVILEGE function, 13-16

E

e-mail alert in rule set, 5-9
enabling system features with Enabled default rule set, 5-2
encrypted information, 16-14
enterprise identities, Enterprise_Identity default factor, 7-3
Enterprise Manager
See Oracle Enterprise Manager
errors
factor error options, 7-10
rule set error options, 5-4
event handler
rule sets, 5-4
examples
DVSYS.DBMS_MACUTL constants, 13-4
realms, 4-10
separation of duty matrix, C-2
See also tutorials
EXECUTE ANY PROCEDURE privilege, securing for external C callouts, C-14
EXECUTE ANY PROCEDURE privilege, securing for Java stored procedures, C-11
Execute Privileges to Strong SYS Packages Report, 16-7
EXECUTE_CATALOG_ROLE role, 16-11
EXEMPT ACCESS POLICY system privilege, 16-10
external C callouts
EXECUTE ANY PROCEDURE privilege, C-14
security considerations, C-14

F

Factor Audit Report, 16-4
Factor Configuration Issues Report, 16-3
Factor Without Identities Report, 16-3
factors
about, 7-1
assignment, 7-8
disabled rule set, 16-3
incomplete rule set, 16-3
validate, 7-8
assignment operation, 16-4
audit events, custom, A-1
audit options, 7-9
child factors
about, 7-6
Factor Configuration Issues Report, 16-3
mapping, 7-13
creating, 7-4
creating names, 7-4
data dictionary views, 7-30
default factors, 7-2
default factors not showing in Database Vault Administrator, D-2
deleting, 7-14
domain, finding with DVF.F$DOMAIN, 14-10
DVSYS.DBMS_MACUTL constants, example of, 13-5
editing, 7-10
error options, 7-10
evaluate, 7-7
evaluation operation, 16-4
factor type
about, 7-5
selecting, 7-5
factor-identity pair mapping, 7-13
functionality, 7-15
functions
DVSYS.DBMS_MACUTL (utility), 13-1 to 13-6
DVSYS.DBMS_MACUTL constants (fields), 13-1
guidelines, 7-28
identifying using child factors, 7-13
identities
about, 7-6, 7-11
adding to factor, 7-10
assigning, 7-7
configuring, 7-11
creating, 7-11
data dictionary views, 7-30
database session, 7-6
deleting, 7-13
determining with DVSYS.GET_FACTOR, 7-6
editing, 7-13
enterprise-wide users, 14-10
how factor identities work, 7-6
labels, 7-7, 7-12
mapping, about, 7-13
mapping, identified, 7-6
mapping, procedure, 7-13
mapping, tutorial, 7-22
Oracle Label Security labels, 7-7
reports, 7-29

resolving, 7-5
retrieval methods, 7-7
setting dynamically, 14-2
trust levels, 7-6, 7-11
with Oracle Label Security, 7-6
initialization, command rules, 6-2
invalid audit options, 16-3
label, 16-3
naming conventions, 7-4
Oracle Virtual Private Database, attaching factors to, 9-6
parent factors, 7-6
performance effect, 7-29
procedures
DVSYS.DBMS_MACADM (configuration), 11-26
process flow, 7-15
propagating policies to other databases, 9-2
reports, 7-29
retrieving, 7-16
retrieving with DVSYS.GET_FACTOR, 14-2
rule sets
selecting, 7-8
used with, 7-1
setting, 7-17
setting with DVSYS.SET_FACTOR, 14-2
troubleshooting
auditing report, 16-4
configuration problems, D-2
tips, D-2
type (category of factor), 7-5
validating, 7-8
values (identities), 7-1
views
DBA_DV_CODE, 10-13
DBA_DV_FACTOR_LINK, 10-16
DBA_DV_FACTOR_TYPE, 10-17
DBA_DV_IDENTITY, 10-17
DBA_DV_IDENTITY_MAP, 10-18
DBA_DV_MAC_POLICY_FACTOR, 10-19
ways to assign, 7-6
See also rule sets
fine-grained auditing
DBMS_RLS package, who can grant EXECUTE on, 2-5
functions
command rules
DVSYS.DBMS_MACUTL (utility), 13-1 to 13-6
DVSYS schema enabling, 14-1
factors
DVSYS.DBMS_MACUTL (utility), 13-1 to 13-6
Oracle Label Security policy
DVSYS.DBMS_MACADM (configuration), 11-48
realms
DVSYS.DBMS_MACUTL (utility), 13-1 to 13-6
rule sets
DVSYS.DBMS_MACADM (configuration), 11-12
DVSYS.DBMS_MACUTL (utility), 13-1 to 13-6
PL/SQL functions for inspecting SQL, 14-14
secure application roles
DVSYS.DBMS_MACADM (configuration), 11-45
DVSYS.DBMS_MACSEC_ROLES (configuration), 12-1
DVSYS.DBMS_MACUTL (utility), 13-1

G

general security reports, 16-5 to 16-11
GRANT statement
monitoring, 15-3
guidelines
ALTER SESSION privilege, C-10
ALTER SYSTEM privilege, C-10
command rules, 6-8
CREATE ANY JOB privilege, C-10
CREATE EXTERNAL JOB privilege, C-10
CREATE JOB privilege, C-10
DBMS_FILE_TRANSFER package, C-8
factors, 7-28
general security, C-1 to C-17
Java stored procedures, C-11
LogMiner packages, C-10
managing DV_OWNER and DV_ACCTMGR accounts, 10-9
operating system access, C-5
Oracle software owner, C-6
performance effect, 7-29
realms, 4-11
recycle bin, C-9
root access, C-5
root user access, C-6
rule sets, 5-18
secure application roles, 8-3
SYSDBA access, C-6
SYSDBA privilege, limiting, C-4
SYSOPER access, C-6
SYSTEM schema and application tables, C-4
SYSTEM user account, C-4
trusted accounts and roles, C-5
using Database Vault in a production environment, C-6
UTL_FILE package, C-8

H

hackers
See security attacks
Hierarchical System Privileges by Database Account Report, 16-7
host names
finding with DVF.F$DATABASE_HOSTNAME, 14-8

I

identities
See factors, identities
Identity Configuration Issues Report, 16-3
IDLE_TIME resource profile, 16-12

incomplete rule set, 16-3
role enablement, 16-4
initialization parameters
Allow System Parameters default rule set, 5-2
modified after installation, 2-1
modified by Oracle Database Vault, 2-1
reports, 16-11 to 16-12
insider threats
See intruders
installations
security considerations, C-7
intruders
See security attacks
IP addresses
Client_IP default factor, 7-2
defined with factors, 7-1

J

Java Policy Grants Report, 16-13
Java stored procedures
EXECUTE ANY PROCEDURE privilege, C-11
guidelines on managing, C-11
realm protections, 4-9

L

Label Security Integration Audit Report, 16-5
labels
about, 7-12
See also Oracle Label Security
languages
consistency between Oracle Database and operating system, D-2
finding with DVF.F$LANG, 14-11
finding with DVF.F$LANGUAGE, 14-12
name
Lang default factor, 7-3
Language default factor, 7-3
LBACSYS account
about, 10-10
auditing policy, A-8
factor integration with OLS policy requirement, 9-8
See also Oracle Label Security
LBACSYS schema
auditing policy, A-9
listener, starting, B-4
locked out accounts, solution for, B-1
log files
database process, 3-5
Database Vault log files, A-2
logging on
reports, Core Database Audit Report, 16-12
LogMiner packages
guidelines, C-10
lsnrctl process, starting, B-4

M

maintenance on Oracle Database Vault, B-1
managing user accounts and profiles on own account, Can Maintain Own Accounts default rule set, 5-2
managing user accounts and profiles, Can Maintain Accounts/Profiles default rule set, 5-2
mapping identities, 7-13
monitoring
activities, 15-1 to 15-4
My Oracle Support, 3-1

N

naming conventions
factors, 7-4
rule sets, 5-3
rules, 5-6
network protocol
finding with DVF.F$NETWORK_PROTOCOL, 14-13
network protocol, Network_Protocol default factor, 7-4
NOAUDIT statement
monitoring, 15-3
Non-Owner Object Trigger Report, 16-14
nonsystem database accounts, 16-6

O

Object Access By PUBLIC Report, 16-6
Object Access Not By PUBLIC Report, 16-6
Object Dependencies Report, 16-6
object owners
nonexistent, 16-3
reports
Command Rule Configuration Issues Report, 16-3
object privilege reports, 16-5 to 16-6
objects
command rule objects
name, 6-5
owner, 6-5
processing, 6-5
dynamic SQL use, 16-13
monitoring, 15-3
object names
finding with DVSYS.DV_DICT_OBJ_NAME, 14-17
object owners
finding with DVSYS.DV_DICT_OBJ_OWNER, 14-16
object privileges
checking with DVSYS.DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE function, 13-6
realms
object name, 4-4
object owner, 4-4
object type, 4-4
procedures for registering, 11-2
reports
Access to Sensitive Objects Report, 16-8
Accounts with SYSDBA/SYSOPER Privilege Report, 16-8
Direct Object Privileges Report, 16-6

Execute Privileges to Strong SYS Packages Report, 16-7
Non-Owner Object Trigger Report, 16-14
Object Access By PUBLIC Report, 16-6
Object Access Not By PUBLIC Report, 16-6
Object Dependencies Report, 16-6
Objects Dependent on Dynamic SQL Report, 16-13
OS Directory Objects Report, 16-13
privilege, 16-5 to 16-6
Public Execute Privilege To SYS PL/SQL Procedures Report, 16-8
sensitive, 16-7 to 16-9
System Privileges By Privilege Report, 16-7
types
finding with DVSYS.DV_DICT_OBJ_TYPE, 14-16
views, DBA_DV_REALM_OBJECT, 10-22
See also database objects
Objects Dependent on Dynamic SQL Report, 16-13
OEM
See Oracle Enterprise Manager (OEM)
OLS
See Oracle Label Security
operating system access
guideline for using with Database Vault, C-5
operating systems
reports
OS Directory Objects Report, 16-13
OS Security Vulnerability Privileges Report, 16-11
vulnerabilities, 16-11
Oracle database
See databases
Oracle Database Vault
about, 1-1
components, 1-2
disabling
checking if disabled, B-2
procedures for, B-1
reasons for, B-1
enabling
checking if enabled, B-2
procedures for, B-1
frequently asked questions, 1-1
integrating with other Oracle products, 9-1
maintenance, B-1
Oracle Database installation, affect on, 2-1
Oracle Database Vault Administrator
starting without Oracle Enterprise Manager, 3-4
Oracle Database Vault Administrator (DVA)
logging on from Oracle Enterprise Manager Database Control, 3-1
logging on from Oracle Enterprise Manager Grid Control, 3-2
logging on without Oracle Enterprise Manager, 3-4
Oracle Database Vault Configuration Assistant (DVCA)
about, 1-3
Oracle Enterprise Manager
DBSNMP account
realm for, 4-2
default realm used for, 4-2
performance tools, 4-13
setting URL for Database Vault Administrator, 9-1
SYSMAN account
realm for, 4-2
Oracle Enterprise Manager Database Control
starting Oracle Database Vault from, 3-1
Oracle Enterprise Manager Grid Control
propagating Database Vault policies to other databases, 9-2
starting Oracle Database Vault from, 3-2
Oracle Enterprise User Security, integrating with Oracle Database Vault, 9-5
Oracle Internet Directory Distinguished Name, Proxy_Enterprise_Identity default factor, 7-4
Oracle Label Security
database option, 1-3
policies
Oracle Policy Manager, 1-3
Oracle Label Security (OLS)
audit events, custom, A-1
checking if installed using DVSYS.DBMS_MACUTL functions, 13-6
data dictionary views, 9-13
functions
DVSYS.DBMS_MACUTL (utility), 13-1
how Database Vault integrates with, 9-6
initialization, command rules, 6-2
integration with Oracle Database Vault
example, 9-9
Label Security Integration Audit Report, 16-5
procedure, 9-7
requirements, 9-7
labels
about, 7-12
determining with GET_FACTOR_LABEL, 14-5
invalid label identities, 16-3
policies
accounts that bypass, 16-10
monitoring policy changes, 15-3
nonexistent, 16-3
procedures
DVSYS.DBMS_MACADM (configuration), 11-48
reports, 9-13
views
DBA_DV_MAC_POLICY, 10-19
DBA_DV_MAC_POLICY_FACTOR, 10-19
DBA_DV_POLICY_LABEL, 10-20
See also LBACSYS account
Oracle MetaLink
See My Oracle Support
Oracle Policy Manager
used with Oracle Label Security, 1-3
Oracle Real Application Clusters

compatibility with Oracle Database Vault, 1-1
multiple factor identities, 7-6
Oracle Recovery Manager (RMAN)
in an Oracle Database Vault environment, 9-14
Oracle software owner, guidelines on managing, C-6
Oracle Technology Network (OTN), xx
Oracle Virtual Private Database
DBMS_RLS package, who can grant EXECUTE on, 2-5
Oracle Virtual Private Database (VPD)
accounts that bypass, 16-10
factors, attaching to, 9-6
GRANT EXECUTE privileges with Grant VPD Administration default rule set, 5-2
using Database Vault factors with Oracle Label Security, 9-9
OS Directory Objects Report, 16-13
OS Security Vulnerability Privileges Report, 16-11
OS_AUTHENT_PREFIX initialization parameter, 2-2
OS_ROLES initialization parameter, 2-2

P

parameters
modified after installation, 2-1
reports
Security Related Database Parameters Report, 16-11
parent factors
See factors
Password History Access Report, 16-11
passwords
forgotten, solution for, B-1
reports, 16-12
Database Account Default Password Report, 16-12
Password History Access Report, 16-11
Username/Password Tables Report, 16-14
patches
security consideration, C-7
two-person integrity used for, 5-14
performance effect
command rules, 6-9
realms, 4-12
reports
Resource Profiles Report, 16-12
System Resource Limits Report, 16-12
rule sets, 5-19
secure application roles, 8-8
performance tools
Database Control, realms, 4-13
Oracle Enterprise Manager
command rules, 6-9
factors, 7-29
realms, 4-13
rule sets, 5-19
secure application roles, 8-8
Oracle Enterprise Manager Database Control
command rules, 6-9
factors, 7-29
rule sets, 5-19
secure application roles, 8-8
STATSPACK utility
command rules, 6-9
factors, 7-29
realms, 4-13
rule sets, 5-19
secure application roles, 8-8
TKPROF utility
command rules, 6-9
factors, 7-29
realms, 4-13
rule sets, 5-19
secure application roles, 8-8
PL/SQL
packages
summarized, 14-18
unwrapped bodies, 16-13
Unwrapped PL/SQL Package Bodies Report, 16-13
PL/SQL factor functions, 14-5
policy changes, monitoring, 15-3, 15-4
port number
finding, 3-4
Oracle Database Vault, 3-4
privileges
ANY privileges, 10-7
checking with DVSYS.DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE function, 13-6
existing users and roles, Database Vault affect on, 2-3
least privilege principle
violations to, 16-13
monitoring
GRANT statement, 15-3
REVOKE statement, 15-3
Oracle Database Vault restricting, 2-3
reports
Accounts With DBA Roles Report, 16-10
ALTER SYSTEM or ALTER SESSION Report, 16-10
ANY System Privileges for Database Accounts Report, 16-7
AUDIT Privileges Report, 16-11
Database Accounts With Catalog Roles Report, 16-11
Direct and Indirect System Privileges By Database Account Report, 16-7
Direct System Privileges By Database Account Report, 16-7
Hierarchical System Privileges By Database Account Report, 16-7
listed, 16-9
OS Directory Objects Report, 16-13
Privileges Distribution By Grantee Report, 16-9
Privileges Distribution By Grantee, Owner Report, 16-9
Privileges Distribution By Grantee, Owner,

Privilege Report, 16-9
WITH ADMIN Privilege Grants Report, 16-10
WITH GRANT Privileges Report, 16-11
roles
checking with DVSYS.DBMS_MACUTL.USER_HAS_ROLE_VARCHAR function, 13-6
system
checking with DVSYS.DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE function, 13-6
views
DBA_DV_PUB_PRIVS, 10-20
DBA_DV_USER_PRIVS, 10-25
DBA_DV_USER_PRIVS_ALL, 10-26
Privileges Distribution By Grantee Report, 16-9
Privileges Distribution By Grantee, Owner Report, 16-9
Privileges Distribution By Grantee, Owner, Privilege Report, 16-9
privileges using external password, 16-8
problems, diagnosing, D-1
procedures
command rules
DVSYS.DBMS_MACADM (configuration), 11-22
factors
DVSYS.DBMS_MACADM (configuration), 11-26
realms
DVSYS.DBMS_MACADM (configuration), 11-1
production environments
guidelines for securing, C-6
profiles, 16-11 to 16-12
PUBLIC access to realms, 4-9
Public Execute Privilege To SYS PL/SQL Procedures Report, 16-8

Q

quotas
tablespace, 16-14

R

RAC
See Oracle Real Application Clusters
Realm Audit Report, 16-4
Realm Authorization Configuration Issues Report, 16-3
realms
about, 4-1
adding roles to as grantees, 4-12
audit events, custom, A-1
authentication-related procedures, 11-2
authorization
how realm authorizations work, 4-9
process flow, 4-9
troubleshooting, D-2
updating with DVSYS.DBMS_MACADM.UPDATE_REALM_AUTH, 11-2
authorizations
grantee, 4-6
rule set, 4-6
creating, 4-2
data dictionary views, 4-13
default realms, 4-2
default realms not showing in Database Vault Administrator, D-2
deleting, 4-7
disabling, 4-7
DV_REALM_OWNER role, 10-5
DV_REALM_RESOURCE role, 10-5
DVSYS.DBMS_MACUTL constants, example of, 13-4
editing, 4-3
effect on other Oracle Database Vault components, 4-11
enabling, 4-7
example, 4-10
functions
DVSYS.DBMS_MACUTL (utility), 13-1 to 13-6
DVSYS.DBMS_MACUTL constants (fields), 13-1
guidelines, 4-11
how realms work, 4-8
Java stored procedures, 4-9
object-related procedures, 11-2
performance effect, 4-12
procedures
DVSYS.DBMS_MACADM (configuration), 11-1
process flow, 4-8
propagating policies to other databases, 9-2
PUBLIC access, 4-9
realm authorizations
about, 4-5
realm secured objects
deleting, 4-5
editing, 4-5
object name, 4-4
object owner, 4-4
object type, 4-4
realm system authorizations
creating, 4-6
deleting, 4-7
editing, 4-7
realm-secured objects, 4-4
reports, 4-13
roles
DV_REALM_OWNER, 10-5
DV_REALM_RESOURCE, 10-5
secured object, 16-3
territory a realm protects, 4-4
troubleshooting, D-2
tutorial, 3-5
updating with DVSYS.DBMS_MACADM.UPDATE_REALM, 11-2
views
DBA_DV_CODE, 10-12
DBA_DV_REALM, 10-21
DBA_DV_REALM_AUTH, 10-22

DBA_DV_REALM_OBJECT, 10-22
See also rule sets
RECOVERY_CATALOG_OWNER role, 16-11
recycle bin, guidelines on managing, C-9
REMOTE_LOGIN_PASSWORDFILE initialization parameter, 2-2
REMOTE_OS_AUTHENT initialization parameter, 2-2
REMOTE_OS_ROLES initialization parameter, 2-2
reporting menu
report results page, 16-2
parameter, 16-2
reports
about, 16-1
Access to Sensitive Objects Report, 16-8
Accounts With DBA Roles Report, 16-10
Accounts with SYSDBA/SYSOPER Privilege Report, 16-8
ALTER SYSTEM or ALTER SESSION Report, 16-10
ANY System Privileges for Database Accounts Report, 16-7
AUDIT Privileges Report, 16-11
auditing, 16-4 to 16-5
BECOME USER Report, 16-10
categories of, 16-1
Command Rule Audit Report, 16-4
Command Rule Configuration Issues Report, 16-3
Core Database Audit Report, 16-12
Core Database Vault Audit Trail Report, 16-5
Database Account Default Password Report, 16-12
Database Account Status Report, 16-12
Database Accounts With Catalog Roles Report, 16-11
Direct and Indirect System Privileges By Database Account Report, 16-7
Direct Object Privileges Report, 16-6
Direct System Privileges By Database Account Report, 16-7
Enterprise Manager Grid Control, 9-5
Execute Privileges to Strong SYS Packages Report, 16-7
Factor Audit Report, 16-4
Factor Configuration Issues Report, 16-3
Factor Without Identities, 16-3
general security, 16-5 to 16-11
Hierarchical System Privileges by Database Account Report, 16-7
Identity Configuration Issues Report, 16-3
Java Policy Grants Report, 16-13
Label Security Integration Audit Report, 16-5
Non-Owner Object Trigger Report, 16-14
Object Access By PUBLIC Report, 16-6
Object Access Not By PUBLIC Report, 16-6
Object Dependencies Report, 16-6
Objects Dependent on Dynamic SQL Report, 16-13
OS Directory Objects Report, 16-13
OS Security Vulnerability Privileges, 16-11
Password History Access Report, 16-11
permissions for running, 16-1
privilege management, 16-9
Privileges Distribution By Grantee Report, 16-9
Privileges Distribution By Grantee, Owner Report, 16-9
Privileges Distribution By Grantee, Owner, Privilege Report, 16-9
Public Execute Privilege To SYS PL/SQL Procedures Report, 16-8
Realm Audit Report, 16-4
Realm Authorization Configuration Issues Report, 16-3
Resource Profiles Report, 16-12
Roles/Accounts That Have a Given Role Report, 16-11
Rule Set Configuration Issues Report, 16-4
running, 16-2
Secure Application Configuration Issues Report, 16-4
Secure Application Role Audit Report, 16-5
Security Policy Exemption Report, 16-10
Security Related Database Parameters, 16-11
security vulnerability, 16-13 to 16-14
System Privileges By Privilege Report, 16-7
System Resource Limits Report, 16-12
Tablespace Quotas Report, 16-14
Unwrapped PL/SQL Package Bodies Report, 16-13
Username /Password Tables Report, 16-14
WITH ADMIN Privileges Grants Report, 16-10
WITH GRANT Privileges Report, 16-11
required parameters page
% wildcard, 16-2
Resource Profiles Report, 16-12
resources
reports
Resource Profiles Report, 16-12
System Resource Limits Report, 16-12
REVOKE statement
monitoring, 15-3
roles
adding to realms as grantees, 4-12
catalog-based, 16-11
Database Vault default roles, 10-2 to 10-9
privileges, checking with DVSYS.DBMS_MACUTL.USER_HAS_ROLE_VARCHAR function, 13-6
role enablement in incomplete rule set, 16-4
role-based system privileges, 16-7
See also secure application roles
Roles/Accounts That Have a Given Role Report, 16-11
root access
guideline for using with Database Vault, C-5
root access, guidelines on managing, C-6
Rule Set Configuration Issues Report, 16-4
rule sets
about, 5-1

adding existing rules, 5-7
audit options, 5-3
command rules
disabled, 16-3
selecting for, 6-5
used with, 6-1
CONNECT role configured incorrectly, solution for, B-1
creating, 5-2
rules in, 5-5
creating names, 5-3
data dictionary views, 5-19
default rule sets, 5-2
default rule sets not showing in Database Vault Administrator, D-2
deleting
rule set, 5-7
rules from, 5-7
disabled for
factor assignment, 16-3
realm authorization, 16-3
DVSYS.DBMS_MACUTL constants, example of, 13-5
editing
rule sets, 5-5
rules in, 5-6
error options, 5-4
evaluation of rules, 5-5
evaluation options, 5-3
event handlers, 5-4
events firing, finding with DVSYS.DV_SYSEVENT, 14-14
factors, selecting for, 7-8
factors, used with, 7-1
fail code, 5-4
fail message, 5-4
functions
DVSYS.DBMS_MACADM (configuration), 11-12
DVSYS.DBMS_MACUTL (utility), 13-1 to 13-6
DVSYS.DBMS_MACUTL constants (fields), 13-1
PL/SQL functions for rule sets, 14-14
guidelines, 5-18
how rule sets work, 5-8
incomplete, 16-3
naming conventions, 5-3
nested rules, 5-8
performance effect, 5-19
procedures
DVSYS.DBMS_MACADM (configuration), 11-12
process flow, 5-8
propagating policies to other databases, 9-2
reports, 5-19
rules that exclude one user, 5-8
troubleshooting, D-2
tutorial, 5-9
views
DBA_DV_RULE, 10-23
DBA_DV_RULE_SET, 10-24
DBA_DV_RULE_SET_RULE, 10-25
See also command rules, factors, realms, rules, secure application roles
rules
about, 5-5
creating, 5-5
creating names, 5-6
data dictionary views, 5-19
deleting from rule set, 5-7
editing, 5-6
existing rules, adding to rule set, 5-7
naming conventions, 5-6
nested within a rule set, 5-8
removing from rule set, 5-7
reports, 5-19
troubleshooting, D-2
views
DBA_DV_RULE, 10-23
DBA_DV_RULE_SET_RULE, 10-25
See also rule sets
rules sets
audit event, custom, A-1

S

schemas
DVF, 10-2
DVSYS, 10-1
Secure Application Configuration Issues Report, 16-4
secure application role, 8-1
Secure Application Role Audit Report, 16-5
secure application roles
audit event, custom, A-1
creating, 8-2
data dictionary view, 8-9
deleting, 8-3
DVSYS.DBMS_MACSEC_ROLES.SET_ROLE function, 8-3
functionality, 8-4
functions
DVSYS.DBMS_MACADM (configuration), 11-45
DVSYS.DBMS_MACSEC_ROLES (configuration), 12-1
DVSYS.DBMS_MACSEC_ROLES package, 12-1
DVSYS.DBMS_MACUTL (utility), 13-1
DVSYS.DBMS_MACUTL constants (fields), 13-1
guidelines on managing, 8-3
performance effect, 8-8
procedure
DVSYS.DBMS_MACADM (configuration), 11-45
procedures and functions
DVSYS.DBMS_MACUTL (utility), 13-5
propagating policies to other databases, 9-2
reports, 8-9
Rule Set Configuration Issues Report, 16-4

troubleshooting, D-2
troubleshooting with auditing report, 16-5
tutorial, 8-4
views
DBA_DV_ROLE, 10-23
See also roles, rule sets
security attacks
Denial of Service (DoS) attacks
finding system resource limits, 16-12
Denial of Service attacks
finding tablespace quotas, 16-14
eliminating audit trail, 16-11
monitoring security violations, 15-1
Oracle Database Vault addressing insider threats, 1-5
reports
AUDIT Privileges Report, 16-11
Objects Dependent on Dynamic SQL Report, 16-13
Privileges Distribution By Grantee, Owner Report, 16-9
Unwrapped PL/SQL Package Bodies Report, 16-14
SQL injection attacks, 16-13
tracking
with factor auditing, 7-10
with rule set auditing, 5-4
security policies
monitoring changes, 15-4
security policies, Oracle Database Vault addressing, 1-5
Security Policy Exemption Report, 16-10
Security Related Database Parameters Report, 16-11
security violations
monitoring attempts, 15-1
security vulnerabilities
how Database Vault addresses, 1-6
operating systems, 16-11
reports, 16-13 to 16-14
Security Related Database Parameters Report, 16-11
root operating system directory, 16-13
SELECT statement
controlling with command rules, 6-1
SELECT_CATALOG_ROLE role, 16-11
sensitive objects reports, 16-7 to 16-9
separation of duty concept
about, C-1
command rules, 6-3
database accounts, 10-10
database accounts, suggested, 10-10
database roles, 2-3
Database Vault Account Manager role, 10-9
documenting tasks, C-3
example matrix, C-2
Oracle Database Vault enforcing, 1-1
realms, 1-6
restricting privileges, 2-3
roles, 10-2
tasks in Oracle Database Vault environment, C-2
sessions
audit events, custom, A-1
DVSYS.DBMS_MACUTL fields, 13-2
finding session user with DVF.F$SESSION_USER, 14-13
restricting data based on, 7-22
retrieving information with functions, 11-26
SQL injection attacks, detecting with Object Dependent on Dynamic SQL Report, 16-13
SQL statements
default command rules that protect, 6-2
SQL text, finding with DVSYS.DV_SQL_TEXT, 14-17
SQL92_SECURITY initialization parameter, 2-3
subfactors
See child factors under factors topic
SYS schema
command rules, 6-5
SYS user account
adding to realm authorization, 4-12
SYS.AUD$ table
location for Oracle Database Vault, 2-5
object owner for realm protection, 4-4
protecting with realm, 2-5
SYSDBA access
guidelines on managing, C-6
SYSDBA privilege
limiting, importance of, C-4
SYS.FGA_LOG$ table
protecting with realm, 2-5
SYSMAN user account
realm for, 4-2
SYSOPER access
guidelines on managing, C-6
system features
disabling with Disabled rule set, 5-2
enabling with Enabled rule set, 5-2
system privileges
checking with DVSYS.DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE function, 13-6
reports
System Privileges By Privileges Report, 16-7
System Privileges By Privilege Report, 16-7
System Resource Limits Report, 16-12
system root access, guideline on managing, C-6
SYSTEM schema
application tables in, C-4
SYSTEM user account
guidelines for using with Database Vault, C-4
SYSTEM.AUD$ table
location for Oracle Database Vault, 2-5

T

tablespace quotas, 16-14
Tablespace Quotas Report, 16-14
third party products, affected by Oracle Database Vault, B-1
time data
DVSYS.DBMS_MACUTL functions, 13-6
trace files

about, D-1
enabling, D-1
Transparent Data Encryption, used with Oracle Database Vault, 9-6
triggers
different from object owner account, 16-14
reports, Non-Owner Object Trigger Report, 16-14
troubleshooting
access security sessions, 16-5
auditing reports, using, 16-4
command rules, D-1
Database Vault Administrator not showing default realms, command rules, rule sets, or factors, D-2
events, D-1
factors, D-2
general diagnostic tips, D-2
locked out accounts, B-1
passwords, forgotten, B-1
realms, D-2
rule sets, D-2
rules, D-2
secure application roles, 16-5
trust levels
about, 7-11
determining for identities with DVSYS.GET_TRUST_LEVEL_FOR_IDENTITY, 14-3
determining with DVSYS.GET_TRUST_LEVEL, 14-3
factor identity, 7-11
factors, 7-11
for factor and identity requested, 14-3
identities, 7-6
of current session identity, 14-3
trusted users
accounts and roles that should be limited, C-5 to C-6
default for Oracle Database Vault, C-5
tutorials
access, granting with secure application roles, 8-4
ad hoc tool access, preventing, 7-17
configuring two-person integrity (TPI), 5-14
Database Vault factors with Virtual Private Database and Oracle Label Security, 9-9
e-mail alert in rule set, 5-9
factors, mapping identities, 7-22
Oracle Label Security integration with Oracle Database Vault, 9-9
restricting access based on session data, 7-22
restricting user activities with command rules, 6-6
schema, protecting with a realm, 3-5
See also examples
two-man rule security
See two-person integrity (TPI)
two-person integrity (TPI)
about, 5-14
configuring with a rule set, 5-14

U

Unwrapped PL/SQL Package Bodies Report, 16-13
user names
reports, Username/Password Tables Report, 16-14
USER_HISTORY$ table, 16-11
Username/Password Tables Report, 16-14
users
enterprise identities, finding with DVF.F$PROXY_ENTERPRISE_IDENTITY, 14-13
enterprise-wide identities, finding with DVF.F$ENTERPRISE_IDENTITY, 14-10
finding session user with DVF.F$SESSION_USER, 14-13
login user name, finding with DVSYS.DV_LOGIN_USER, 14-15
restricting access by factor identity, 7-22
utility functions
See DVSYS.DBMS_MACUTL package
UTL_FILE object, 16-6
UTL_FILE package, guidelines on managing, C-8

V

views
Oracle Database Vault-specific views, 10-11 to 10-26
See also names beginning with DBA_DV
VPD
See Oracle Virtual Private Database (VPD)

W

wildcard, %, 16-2
WITH ADMIN Privileges Grants Report, 16-10
WITH ADMIN status, 16-7
WITH GRANT clause, 16-11
WITH GRANT Privileges Report, 16-11