Oracle® Database Vault

Administrator's Guide

23c
F46691-07
October 2023

Oracle Database Vault Administrator's Guide, 23c

F46691-07

Copyright © 1996, 2023, Oracle and/or its affiliates.

Primary Author: Patricia Huey

Contributors: Taousif Ansari , Tom Best, Ji-won Byun, Martin Cheng, Chi Ching Chui, Richard Evans, Scott Gaetjen, Viksit Gaur, Rishabh Gupta, Lijie Heng, Suhas Javagal , Dominique Jeunot, Peter Knaggs, Suman Kumar, Rudregowda Mallegowda, Yi Ouyang, Hozefa Palitanawala, Gayathri Sairamkrishnan, Vipin Samar, James Spiller, Srividya Tata, Kamal Tbeileh, Saravana Soundararajan, Sudheesh Varma, Peter Wahl, Alan Williams

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

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

If this is software, software documentation, data (as defined in the Federal Acquisition Regulation), or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software," "commercial computer software documentation," or "limited rights data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed, or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

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

Oracle®, Java, and MySQL are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience xxv
Documentation Accessibility xxv
Diversity and Inclusion xxv
Related Documents xxv
Conventions xxvi

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

Changes in Oracle Database Vault 23c xxvii

1 Introduction to Oracle Database Vault

1.1 What Is Oracle Database Vault? 1-1
1.1.1 About Oracle Database Vault 1-2
1.1.2 Controls for Privileged Accounts 1-2
1.1.3 Controls for Database Configuration 1-3
1.1.4 Enterprise Applications Protection Policies 1-3
1.2 What Privileges Do You Need to Use Oracle Database Vault? 1-4
1.3 Components of Oracle Database Vault 1-4
1.3.1 Oracle Database Vault Access Control Components 1-5
1.3.2 Oracle Database Vault DVSYS and DVF Schemas 1-6
1.3.3 Oracle Database Vault PL/SQL Interfaces and Packages 1-6
1.3.4 Oracle Database Vault Reporting and Monitoring Tools 1-6
1.3.5 Oracle Enterprise Manager Cloud Control Database Vault Administrator Pages 1-7
1.4 How Oracle Database Vault Addresses Compliance Regulations 1-7
1.5 How Oracle Database Vault Protects Privileged User Accounts 1-8
1.6 How Oracle Database Vault Allows for Flexible Security Policies 1-8
1.7 How Oracle Database Vault Addresses Database Consolidation Concerns 1-9
1.8 How Oracle Database Vault Works in a Multitenant Environment 1-10

2 What to Expect After You Enable Oracle Database Vault

2.1 Initialization and Password Parameter Settings That Change 2-1
2.2 How Oracle Database Vault Restricts User Authorizations 2-2
2.3 Oracle Database Vault-Specific Database Roles to Enforce Separation of Duties 2-3
2.4 Privileges That Are Revoked from Existing Users and Roles 2-3
2.5 Privileges That Are Prevented for Existing Users and Roles 2-5

3 Getting Started with Oracle Database Vault

3.1 About Configuring and Enabling Oracle Database Vault in Oracle Database 3-1
3.2 Configuring and Enabling Oracle Database Vault 3-2
3.2.1 About Configuring and Enabling Database Vault 3-3
3.2.2 Configuring and Enabling Database Vault in the CDB Root 3-3
3.2.3 Configuring and Enabling Database Vault Common Users to Manage Specific PDBs 3-6
3.2.4 Configuring and Enabling Database Vault Local Users to Manage Specific PDBs 3-8
3.2.5 Configuring and Enabling Oracle Database Vault in an Oracle Real Application Clusters Environment 3-11
3.2.6 Creating a Profile to Protect the DV_OWNER and DV_ACCTMGR Users 3-12
3.2.7 Manually Installing Oracle Database Vault 3-13
3.3 Verifying That Database Vault Is Configured and Enabled 3-14
3.4 Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control 3-15
3.5 Quick Start Tutorial: Securing a Schema from DBA Access 3-16
3.5.1 About This Tutorial 3-17
3.5.2 Step 1: Log On as SYSTEM to Access the HR Schema 3-17
3.5.3 Step 2: Create a Realm 3-18
3.5.4 Step 3: Create a Unified Audit Policy for Realm Violations 3-19
3.5.5 Step 4: Create the SEBASTIAN User Account 3-20
3.5.6 Step 5: Have User SEBASTIAN Test the Realm 3-20
3.5.7 Step 6: Create an Authorization for the Realm 3-21
3.5.8 Step 7: Test the Realm 3-21
3.5.9 Step 8: View Audit Records from Realm Violations 3-22
3.5.10 Step 9: Remove the Components for This Tutorial 3-23

4 Configuring Realms

4.1 What Are Realms? 4-2
4.1.1 About Realms 4-2
4.1.2 Mandatory Realms to Restrict User Access to Objects within a Realm 4-3
4.1.3 Realms in a Multitenant Environment 4-4

4.1.4 Object Types That Realms Can Protect 4-5
4.2 Default Realms 4-5
4.2.1 Oracle Database Vault Realm 4-6
4.2.2 Database Vault Account Management Realm 4-6
4.2.3 Oracle Enterprise Manager Realm 4-7
4.2.4 Oracle Default Schema Protection Realm 4-7
4.2.5 Oracle System Privilege and Role Management Realm 4-8
4.2.6 Oracle Default Component Protection Realm 4-9
4.2.7 Oracle Label Security Realm 4-9
4.2.8 Oracle GoldenGate Protection Realm 4-9
4.2.9 Oracle Audit Realm 4-10
4.3 Creating a Realm 4-10
4.4 Modifying a Realm 4-13
4.5 Deleting a Realm 4-14
4.6 About Realm-Secured Objects 4-15
4.7 About Realm Authorization 4-16
4.8 Realm Authorizations in a Multitenant Environment 4-16
4.9 How Realms Work 4-17
4.10 How Authorizations Work in a Realm 4-19
4.10.1 About Authorizations in a Realm 4-19
4.10.2 Examples of Realm Authorizations 4-19
4.10.2.1 Example: Unauthorized User Trying to Create a Table 4-20
4.10.2.2 Example: Unauthorized User Trying to Use the DELETE ANY TABLE Privilege 4-20
4.10.2.3 Example: Authorized User Performing DELETE Operation 4-20
4.11 Access to Objects That Are Protected by a Realm 4-21
4.12 Example of How Realms Work 4-21
4.13 How Realms Affect Other Oracle Database Vault Components 4-22
4.14 Guidelines for Designing Realms 4-22
4.15 How Realms Affect Performance 4-24
4.16 Realm Related Reports and Data Dictionary Views 4-24

5 Configuring Rule Sets

5.1 What Are Rule Sets? 5-1
5.2 Rule Sets and Rules in a Multitenant Environment 5-2
5.3 Default Rules and Rule Sets from Releases Earlier Than Release 12.2 5-2
5.4 Default Rule Sets 5-3
5.5 Creating a Rule Set 5-4
5.6 Creating a Rule to Add to a Rule Set 5-6
5.6.1 What Are Rules? 5-7
5.6.2 Default Rules 5-8

5.6.3 Creating a Rule 5-9
5.6.4 Adding Existing Rules to a Rule Set 5-11
5.6.5 Modifying a Rule Set 5-12
5.6.6 Removing a Rule from a Rule Set 5-12
5.7 Modifying a Rule Set 5-13
5.8 Deleting a Rule Set 5-14
5.9 How Rule Sets Work 5-15
5.9.1 How Oracle Database Vault Evaluates Rules 5-15
5.9.2 Nested Rules within a Rule Set 5-15
5.9.3 Creating Rules to Apply to Everyone Except One User 5-16
5.10 Tutorial: Configuring Two-Person Integrity, or Dual Key Security 5-16
5.10.1 About This Tutorial 5-16
5.10.2 Step 1: Create Users for This Tutorial 5-17
5.10.3 Step 2: Create a Function to Check if User patch_boss Is Logged In 5-17
5.10.4 Step 3: Create Rules, a Rule Set, and a Command Rule to Control User Access 5-18
5.10.5 Step 4: Test the Users' Access 5-19
5.10.6 Step 5: Remove the Components for This Tutorial 5-20
5.11 Guidelines for Designing Rule Sets 5-21
5.12 How Rule Sets Affect Performance 5-22
5.13 Rule Set and Rule Related Reports and Data Dictionary Views 5-22

6 Configuring Command Rules

6.1 What Are Command Rules? 6-1
6.1.1 About Command Rules 6-2
6.1.2 Command Rules in a Multitenant Environment 6-3
6.1.3 Types of Command Rules 6-3
6.1.3.1 CONNECT Command Rule 6-4
6.1.3.2 ALTER SESSION and ALTER SYSTEM Command Rules 6-4
6.2 Default Command Rules 6-6
6.3 SQL Statements That Can Be Protected by Command Rules 6-7
6.4 Creating a Command Rule 6-9
6.5 Modifying a Command Rule 6-10
6.6 Deleting a Command Rule 6-11
6.7 How Command Rules Work 6-12
6.8 Tutorial: Using a Command Rule to Control Table Creations by a User 6-13
6.8.1 Step 1: Create a Table 6-13
6.8.2 Step 2: Create a Command Rule 6-14
6.8.3 Step 3: Test the Command Rule 6-14
6.8.4 Step 4: Remove the Components for This Tutorial 6-15
6.9 Guidelines for Designing Command Rules 6-15

6.10 How Command Rules Affect Performance 6-16
6.11 Command Rule Related Reports and Data Dictionary View 6-16

7 Configuring Factors

7.1 What Are Factors? 7-1
7.2 Default Factors 7-2
7.3 Creating a Factor 7-5
7.4 Adding an Identity to a Factor 7-8
7.4.1 About Factor Identities 7-9
7.4.2 How Factor Identities Work 7-9
7.4.3 About Trust Levels 7-10
7.4.4 About Label Identities 7-11
7.4.5 Creating and Configuring a Factor Identity 7-12
7.4.6 Using Identity Mapping to Configure an Identity to Use Other Factors 7-13
7.4.6.1 About Identity Mapping 7-13
7.4.6.2 Mapping an Identity to a Factor 7-13
7.4.6.3 Deleting an Identity Map 7-15
7.4.7 Modifying a Factor Identity 7-15
7.4.8 Deleting a Factor Identity 7-16
7.5 Modifying a Factor 7-16
7.6 Deleting a Factor 7-17
7.7 How Factors Work 7-18
7.7.1 How Factors Are Processed When a Session Is Established 7-19
7.7.2 How Retrieval Methods Work 7-20
7.7.3 How Factors Are Retrieved 7-21
7.7.4 How Factors Are Set 7-21
7.7.5 How Factor Auditing Works 7-22
7.8 Tutorial: Preventing Ad Hoc Tool Access to the Database 7-22
7.8.1 About This Tutorial 7-23
7.8.2 Step 1: Enable the HR User Accounts 7-23
7.8.3 Step 2: Create the Factor 7-23
7.8.4 Step 3: Create the Rule Set and Rule 7-25
7.8.5 Step 4: Create the CONNECT Command Rule 7-26
7.8.6 Step 5: Test the Ad Hoc Tool Access Restriction 7-26
7.8.7 Step 6: Remove the Components for This Tutorial 7-27
7.9 Guidelines for Designing Factors 7-28
7.10 How Factors Affect Performance 7-29
7.11 Factor Related Reports and Data Dictionary Views 7-29

8 Configuring Secure Application Roles for Oracle Database Vault

8.1 What Are Secure Application Roles in Oracle Database Vault? 8-1
8.2 Security for Oracle Database Vault Secure Application Roles 8-2
8.3 Creating an Oracle Database Vault Secure Application Role 8-2
8.4 Enabling Oracle Database Secure Application Roles to Work with Oracle Database Vault 8-4
8.5 Modifying a Secure Application Role 8-4
8.6 Deleting an Oracle Database Vault Secure Application Role 8-5
8.7 How Oracle Database Vault Secure Application Roles Work 8-5
8.8 Tutorial: Granting Access with Database Vault Secure Application Roles 8-6
8.8.1 About This Tutorial 8-6
8.8.2 Step 1: Create Users for This Tutorial 8-7
8.8.3 Step 2: Enable the OE User Account 8-7
8.8.4 Step 3: Create the Rule Set and Its Rules 8-7
8.8.5 Step 4: Create the Database Vault Secure Application Role 8-8
8.8.6 Step 5: Grant the SELECT Privilege to the Secure Application Role 8-8
8.8.7 Step 6: Test the Database Vault Secure Application Role 8-9
8.8.8 Step 7: Remove the Components for This Tutorial 8-9
8.9 How Secure Application Roles Affect Performance 8-10
8.10 Secure Application Role Related Reports and Data Dictionary View 8-10

9 Configuring Oracle Database Vault Policies

9.1 What Are Database Vault Policies? 9-1
9.1.1 About Oracle Database Vault Policies 9-1
9.1.2 Oracle Database Vault Policies in a Multitenant Environment 9-3
9.2 Default Oracle Database Vault Policies 9-3
9.3 Creating an Oracle Database Policy 9-3
9.4 Modifying an Oracle Database Vault Policy 9-6
9.5 Deleting an Oracle Database Vault Policy 9-6
9.6 Oracle Database Vault Policies Data Dictionary Views 9-7

10 Using Simulation Mode for Logging Realm and Command Rule Activities

10.1 About Simulation Mode 10-1
10.2 Simulation Mode Use Cases 10-2
10.3 Logging Realms in Simulation Mode 10-3
10.3.1 Considerations When Logging Realms in Simulation Mode 10-4
10.3.2 Use Case: All New Realms in Simulation Mode 10-4
10.3.3 Use Case: New Realms Introduced to Existing Realms 10-5
10.3.4 Use Case: Testing the Addition of New Objects in a Realm 10-7

10.3.5 Use Case: Testing the Removal of Objects from a Realm 10-7
10.3.6 Use Case: Testing the Addition of an Authorized User to a Realm 10-7
10.3.7 Use Case: Testing the Removal of an Authorized User from a Realm 10-7
10.3.8 Use Case: Testing New Factors with Realms 10-8
10.3.9 Use Case: Testing Changes to an Existing Command Rule 10-8
10.4 Tutorial: Tracking Violations to a Realm Using Simulation Mode 10-8
10.4.1 About This Tutorial 10-9
10.4.2 Step 1: Create Users for This Tutorial 10-9
10.4.3 Step 2: Create a Realm and an Oracle Database Vault Policy 10-10
10.4.4 Step 3: Test the Realm and Policy 10-11
10.4.5 Step 4: Query the DBA_DV_SIMULATION_LOG View for Violations 10-12
10.4.6 Step 5: Enable and Re-test the Realm 10-13
10.4.7 Step 6: Remove the Components for This Tutorial 10-13

11 Integrating Oracle Database Vault with Other Oracle Products

11.1 Integrating Oracle Database Vault with Transparent Data Encryption 11-1
11.2 Attaching Factors to an Oracle Virtual Private Database 11-2
11.3 Integrating Oracle Database Vault with Oracle Label Security 11-2
11.3.1 How Oracle Database Vault Is Integrated with Oracle Label Security 11-3
11.3.2 Requirements for Using Oracle Database Vault with Oracle Label Security 11-3
11.3.3 Using Oracle Database Vault Factors with Oracle Label Security Policies 11-4
11.3.3.1 About Using Oracle Database Vault Factors with Oracle Label Security Policies 11-4
11.3.3.2 Configuring Factors to Work with an Oracle Label Security Policy 11-5
11.3.4 Tutorial: Integrating Oracle Database Vault with Oracle Label Security 11-7
11.3.4.1 About This Tutorial 11-7
11.3.4.2 Step 1: Create Users for This Tutorial 11-7
11.3.4.3 Step 2: Create the Oracle Label Security Policy 11-8
11.3.4.4 Step 3: Create Oracle Database Vault Rules to Control the OLS Authorization 11-8
11.3.4.5 Step 4: Update the ALTER SYSTEM Command Rule to Use the Rule Set 11-9
11.3.4.6 Step 5: Test the Authorizations 11-9
11.3.4.7 Step 6: Remove the Components for This Tutorial 11-10
11.3.5 Related Reports and Data Dictionary Views 11-10
11.4 Integrating Oracle Database Vault with Oracle Data Guard 11-11
11.4.1 Step 1: Configure the Primary Database 11-12
11.4.2 Step 2: Configure the Standby Database 11-12
11.4.3 How Auditing Works After an Oracle Database Vault-Oracle Active Data Guard Integration 11-13
11.4.4 Disabling Oracle Database Vault in an Oracle Data Guard Environment 11-13

11.5 Configuring Oracle Internet Directory Using Oracle Database Configuration Asssitant 11-14
11.6 Integrating Oracle Database Vault with Enterprise User Security 11-14
11.6.1 About Integrating Oracle Database Vault with Enterprise User Security 11-14
11.6.2 Configuring an Enterprise User Authorization 11-15
11.6.3 Configuring Oracle Database Vault Accounts as Enterprise User Accounts 11-16
11.7 Integrating Oracle Database Vault with Oracle APEX 11-17
11.7.1 About Integrating Oracle Database Vault with Oracle APEX 11-18
11.7.2 Installing or Upgrading Oracle APEX with Oracle Database Vault Enabled 11-18
11.7.3 Authorizing the Oracle APEX Schema for Oracle Database Vault Activities 11-19
11.7.4 Authorizing Oracle APEX to Use Oracle Scheduler 11-20
11.7.5 Authorizing Oracle APEX to Perform DDL Tasks 11-20
11.7.6 Authorizing Oracle APEX to Perform Information Lifecycle Maintenance Tasks 11-21
11.7.7 Authorizing Oracle APEX to Proxy Users for Oracle Rest Data Services 11-21
11.7.8 Oracle APEX and Application Objects Protected by Oracle Database Vault 11-22
11.7.9 Troubleshooting the Oracle APEX and Database Vault Integration 11-22

12 DBA Operations in an Oracle Database Vault Environment

12.1 Performing DDL Operations in Oracle Database Vault 12-2
12.1.1 Restrictions on Performing DDL Operations in Oracle Database Vault 12-2
12.1.2 Impact of the DV_PATCH_ADMIN Role on DDL Operations 12-3
12.1.3 Impact of Upgrades from Releases 21c and Earlier on DDL Operations 12-3
12.1.4 Impact of the Removal of the DDL Default Authorization of ('%', '%') 12-4
12.2 Using SQL Firewall with Oracle Database Vault 12-4
12.2.1 About Using SQL Firewall with Oracle Database Vault 12-5
12.2.2 Authorizing Users for SQL Firewall in Database Vault 12-5
12.2.3 Revoking SQL Firewall Authorization from Users 12-6
12.3 Using Oracle Database Vault with Oracle Enterprise Manager 12-7
12.3.1 Propagating Oracle Database Vault Configurations to Other Databases 12-7
12.3.2 Enterprise Manager Cloud Control Alerts for Oracle Database Vault Policies 12-9
12.3.3 Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control 12-9
12.4 Using Oracle Data Pump with Oracle Database Vault 12-10
12.4.1 About Using Oracle Data Pump with Oracle Database Vault 12-10
12.4.2 Authorizing Users or Roles for Data Pump Regular Export and Import Operations 12-10
12.4.2.1 About Authorizing Users or Roles for Oracle Data Pump Regular Operations 12-11
12.4.2.2 Levels of Database Vault Authorization for Oracle Data Pump Regular Operations 12-11
12.4.2.3 Authorizing Users or Roles for Oracle Data Pump Regular Operations in Database Vault 12-12
12.4.2.4 Revoking Oracle Data Pump Authorization from Users or Roles 12-13

12.4.3 Authorizing Users or Roles for Data Pump Transportable Export and Import Operations 12-14
12.4.3.1 About Authorizing Users for Oracle Data Pump Transportable Operations 12-15
12.4.3.2 Levels of Database Vault Authorization for Data Pump Transportable Operations 12-15
12.4.3.3 Authorizing Users or Roles for Data Pump Transportable Operations in Database Vault 12-16
12.4.3.4 Revoking Transportable Tablespace Authorization from Users or Roles 12-17
12.4.4 Guidelines for Exporting or Importing Data in a Database Vault Environment 12-18
12.5 Using Oracle Scheduler with Oracle Database Vault 12-19
12.5.1 About Using Oracle Scheduler with Oracle Database Vault 12-19
12.5.2 Granting a Job Scheduling Administrator Authorization for Database Vault 12-20
12.5.3 Revoking Authorization from Job Scheduling Administrators 12-21
12.6 Using Information Lifecycle Management with Oracle Database Vault 12-21
12.6.1 About Using Information Lifecycle Management with Oracle Database Vault 12-22
12.6.2 Authorizing Users for ILM Operations in Database Vault 12-22
12.6.3 Revoking Information Lifecycle Management Authorization from Users 12-23
12.7 Using Oracle Database Replay with Oracle Database Vault 12-23
12.7.1 About Using Database Replay with Oracle Database Vault 12-24
12.7.2 Authorizing Users for Database Replay Operations 12-24
12.7.2.1 Authorizing Users for Workload Capture Operations 12-24
12.7.2.2 Authorizing Users for Workload Replay Operations 12-24
12.7.3 Revoking Database Replay Authorization from Users 12-25
12.7.3.1 Revoking Workload Capture Privileges 12-25
12.7.3.2 Revoking Workload Replay Privileges 12-26
12.8 Running Preprocessor Programs with Oracle Database Vault 12-26
12.8.1 About Running Preprocessor Programs with Oracle Database Vault 12-26
12.8.2 Authorizing Users to Run Preprocessor Programs 12-27
12.8.3 Revoking Authorization to Run Execute Preprocessor Programs from Users 12-27
12.9 Using Database Vault Operations Control to Restrict Multitenant Common User Access to Local PDB Data 12-27
12.9.1 About Using Database Vault Operations Control 12-28
12.9.2 How the Addition of Common Users and Packages to an Exception List Works 12-28
12.9.3 Enabling Database Vault Operations Control 12-29
12.9.4 Adding Common Users and Packages to an Exception List 12-30
12.9.5 Deleting Common Users and Packages from an Exception List 12-30
12.9.6 Disabling Database Vault Operations Control 12-31
12.10 Preventing Multitenant Local Users from Blocking Common Operations 12-31
12.10.1 About Preventing Multitenant Local Users from Blocking Common Operations 12-31
12.10.2 Preventing Local Users from Blocking Common Operations 12-32
12.11 Oracle Recovery Manager and Oracle Database Vault 12-32

12.12 Privileges for Using XStream with Oracle Database Vault 12-33
12.13 Privileges for Using Oracle GoldenGate with Oracle Database Vault 12-33
12.14 Using Data Masking in an Oracle Database Vault Environment 12-34
12.14.1 About Data Masking in an Oracle Database Vault Enabled Database 12-35
12.14.2 Adding Data Masking Users to the Data Dictionary Realm Authorizations 12-35
12.14.3 Giving Users Access to Tables or Schemas That They Want to Mask 12-36
12.14.4 Creating a Command Rule to Control Data Masking Privileges 12-36
12.15 Using Oracle Database Auditing with Oracle Database Vault 12-37
12.15.1 About Using Oracle Database Auditing with Oracle Database Vault 12-38
12.15.2 Authorizing the Management of Audit Policies and Related Auditing Tasks 12-38
12.15.2.1 Granting AUDIT_ADMIN Authorization 12-39
12.15.2.2 Revoking AUDIT_ADMIN Authorization 12-39
12.15.3 Authorizing the Ability to View and Analyze Audit Data 12-40
12.15.3.1 Granting AUDIT_VIEWER Authorization 12-40
12.15.3.2 Revoking AUDIT_VIEWER Authorization 12-41
12.16 Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB 12-41
12.17 Using the ORADEBUG Utility with Oracle Database Vault 12-44
12.18 Performing Patch Operations in an Oracle Database Vault Environment 12-44

13 Oracle Database Vault Schemas, Roles, and Accounts

13.1 Oracle Database Vault Schemas 13-1
13.1.1 DVSYS Schema 13-1
13.1.2 DVF Schema 13-3
13.2 Oracle Database Vault Roles 13-3
13.2.1 About Oracle Database Vault Roles 13-4
13.2.2 Privileges of Oracle Database Vault Roles 13-5
13.2.3 Granting Oracle Database Vault Roles to Users 13-9
13.2.4 DV_ACCTMGR Database Vault Account Manager Role 13-10
13.2.5 DV_ADMIN Database Vault Configuration Administrator Role 13-12
13.2.6 DV_AUDIT_CLEANUP Audit Trail Cleanup Role 13-13
13.2.7 DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role 13-14
13.2.8 DV_GOLDENGATE_ADMIN GoldenGate Administrative Role 13-15
13.2.9 DV_GOLDENGATE_REDO_ACCESS GoldenGate Redo Log Role 13-15
13.2.10 DV_MONITOR Database Vault Monitoring Role 13-16
13.2.11 DV_OWNER Database Vault Owner Role 13-17
13.2.12 DV_PATCH_ADMIN Database Vault Database Patch Role 13-19
13.2.13 DV_POLICY_OWNER Database Vault Owner Role 13-20
13.2.14 DV_SECANALYST Database Vault Security Analyst Role 13-21
13.2.15 DV_XSTREAM_ADMIN XStream Administrative Role 13-22
13.3 Oracle Database Vault Accounts Created During Registration 13-23

13.3.1 About Oracle Database Vault Accounts Created During Registration 13-23
13.3.2 Database Accounts Used by Oracle Database Vault 13-23
13.3.3 Model Oracle Database Vault Database Accounts 13-24
13.4 Backup Oracle Database Vault Accounts 13-25

14 Oracle Database Vault Realm APIs

14.1 ADD_AUTH_TO_REALM Procedure 14-1
14.2 ADD_OBJECT_TO_REALM Procedure 14-4
14.3 CREATE_REALM Procedure 14-5
14.4 DELETE_AUTH_FROM_REALM Procedure 14-7
14.5 DELETE_OBJECT_FROM_REALM Procedure 14-8
14.6 DELETE_REALM Procedure 14-9
14.7 DELETE_REALM_CASCADE Procedure 14-10
14.8 RENAME_REALM Procedure 14-10
14.9 UPDATE_REALM Procedure 14-11
14.10 UPDATE_REALM_AUTH Procedure 14-13

15 Oracle Database Vault Rule Set APIs

15.1 DBMS_MACADM Rule Set Procedures 15-1
15.1.1 ADD_RULE_TO_RULE_SET Procedure 15-2
15.1.2 CREATE_RULE Procedure 15-3
15.1.3 CREATE_RULE_SET Procedure 15-5
15.1.4 DELETE_RULE Procedure 15-8
15.1.5 DELETE_RULE_FROM_RULE_SET Procedure 15-8
15.1.6 DELETE_RULE_SET Procedure 15-9
15.1.7 RENAME_RULE Procedure 15-9
15.1.8 RENAME_RULE_SET Procedure 15-10
15.1.9 UPDATE_RULE Procedure 15-11
15.1.10 UPDATE_RULE_SET Procedure 15-12
15.2 Oracle Database Vault PL/SQL Rule Set Functions 15-14
15.2.1 DV_SYSEVENT Function 15-15
15.2.2 DV_LOGIN_USER Function 15-15
15.2.3 DV_INSTANCE_NUM Function 15-16
15.2.4 DV_DATABASE_NAME Function 15-16
15.2.5 DV_DICT_OBJ_TYPE Function 15-16
15.2.6 DV_DICT_OBJ_OWNER Function 15-17
15.2.7 DV_DICT_OBJ_NAME Function 15-17
15.2.8 DV_SQL_TEXT Function 15-18

16 Oracle Database Vault Command Rule APIs

16.1 CREATE_COMMAND_RULE Procedure 16-2
16.2 CREATE_CONNECT_COMMAND_RULE Procedure 16-9
16.3 CREATE_SESSION_EVENT_CMD_RULE Procedure 16-11
16.4 CREATE_SYSTEM_EVENT_CMD_RULE Procedure 16-13
16.5 DELETE_COMMAND_RULE Procedure 16-14
16.6 DELETE_CONNECT_COMMAND_RULE Procedure 16-16
16.7 DELETE_SESSION_EVENT_CMD_RULE Procedure 16-17
16.8 DELETE_SYSTEM_EVENT_CMD_RULE Procedure 16-18
16.9 UPDATE_COMMAND_RULE Procedure 16-19
16.10 UPDATE_CONNECT_COMMAND_RULE Procedure 16-21
16.11 UPDATE_SESSION_EVENT_CMD_RULE Procedure 16-23
16.12 UPDATE_SYSTEM_EVENT_CMD_RULE Procedure 16-24

17 Oracle Database Vault Factor APIs

17.1 DBMS_MACADM Factor Procedures and Functions 17-1
17.1.1 ADD_FACTOR_LINK Procedure 17-3
17.1.2 ADD_POLICY_FACTOR Procedure 17-4
17.1.3 CHANGE_IDENTITY_FACTOR Procedure 17-4
17.1.4 CHANGE_IDENTITY_VALUE Procedure 17-5
17.1.5 CREATE_DOMAIN_IDENTITY Procedure 17-6
17.1.6 CREATE_FACTOR Procedure 17-7
17.1.7 CREATE_FACTOR_TYPE Procedure 17-9
17.1.8 CREATE_IDENTITY Procedure 17-9
17.1.9 CREATE_IDENTITY_MAP Procedure 17-10
17.1.10 DELETE_FACTOR Procedure 17-11
17.1.11 DELETE_FACTOR_LINK Procedure 17-12
17.1.12 DELETE_FACTOR_TYPE Procedure 17-12
17.1.13 DELETE_IDENTITY Procedure 17-13
17.1.14 DELETE_IDENTITY_MAP Procedure 17-13
17.1.15 DROP_DOMAIN_IDENTITY Procedure 17-14
17.1.16 GET_SESSION_INFO Function 17-15
17.1.17 GET_INSTANCE_INFO Function 17-16
17.1.18 RENAME_FACTOR Procedure 17-16
17.1.19 RENAME_FACTOR_TYPE Procedure 17-17
17.1.20 UPDATE_FACTOR Procedure 17-17
17.1.21 UPDATE_FACTOR_TYPE Procedure 17-20
17.1.22 UPDATE_IDENTITY Procedure 17-20
17.2 Oracle Database Vault Run-Time PL/SQL Procedures and Functions 17-21
17.2.1 About Oracle Database Vault Run-Tine PL/SQL Procedures and Functions 17-22

17.2.2 SET_FACTOR Procedure 17-22
17.2.3 GET_FACTOR Function 17-23
17.2.4 GET_FACTOR_LABEL Function 17-23
17.2.5 GET_TRUST_LEVEL Function 17-24
17.2.6 GET_TRUST_LEVEL_FOR_IDENTITY Function 17-25
17.2.7 ROLE_IS_ENABLED Function 17-26
17.3 Oracle Database Vault DVF PL/SQL Factor Functions 17-26
17.3.1 About Oracle Database Vault DVF PL/SQL Factor Functions 17-28
17.3.2 F$AUTHENTICATION_METHOD Function 17-29
17.3.3 F$CLIENT_IP Function 17-29
17.3.4 F$DATABASE_DOMAIN Function 17-30
17.3.5 F$DATABASE_HOSTNAME Function 17-30
17.3.6 F$DATABASE_INSTANCE Function 17-31
17.3.7 F$DATABASE_IP Function 17-31
17.3.8 F$DATABASE_NAME Function 17-31
17.3.9 F$DOMAIN Function 17-32
17.3.10 FDV_CLIENT_IDENTIFIER Function 17-32
17.3.11 FDV_DBLINK_INFO Function 17-33
17.3.12 FDV_MODULE Function 17-33
17.3.13 F$ENTERPRISE_IDENTITY Function 17-34
17.3.14 F$IDENTIFICATION_TYPE Function 17-34
17.3.15 F$LANG Function 17-35
17.3.16 F$LANGUAGE Function 17-35
17.3.17 F$MACHINE Function 17-36
17.3.18 F$NETWORK_PROTOCOL Function 17-36
17.3.19 F$PROXY_ENTERPRISE_IDENTITY Function 17-37
17.3.20 F$PROXY_USER Function 17-37
17.3.21 F$SESSION_USER Function 17-38

18 Oracle Database Vault Secure Application Role APIs

18.1 DBMS_MACADM Secure Application Role Procedures 18-1
18.1.1 CREATE_ROLE Procedure 18-1
18.1.2 DELETE_ROLE Procedure 18-2
18.1.3 RENAME_ROLE Procedure 18-3
18.1.4 UPDATE_ROLE Procedure 18-3
18.2 DBMS_MACSEC_ROLES Secure Application Role Procedure and Function 18-4
18.2.1 CAN_SET_ROLE Function 18-4
18.2.2 SET_ROLE Procedure 18-5

19 Oracle Database Vault Oracle Label Security APIs

19.1 CREATE_MAC_POLICY Procedure 19-1
19.2 CREATE_POLICY_LABEL Procedure 19-3
19.3 DELETE_MAC_POLICY_CASCADE Procedure 19-4
19.4 DELETE_POLICY_FACTOR Procedure 19-4
19.5 DELETE_POLICY_LABEL Procedure 19-5
19.6 UPDATE_MAC_POLICY Procedure 19-6

20 Oracle Database Vault Utility APIs

20.1 DBMS_MACUTL Constants 20-1
20.1.1 DBMS_MACUTL Listing of Constants 20-1
20.1.2 Example: Creating a Realm Using DBMS_MACUTL Constants 20-5
20.1.3 Example: Creating a Rule Set Using DBMS_MACUTL Constants 20-6
20.1.4 Example: Creating a Factor Using DBMS_MACUTL Constants 20-6
20.2 DBMS_MACUTL Package Procedures and Functions 20-7
20.2.1 CHECK_DVSYS_DML_ALLOWED Procedure 20-8
20.2.2 GET_CODE_VALUE Function 20-9
20.2.3 GET_SECOND Function 20-9
20.2.4 GET_MINUTE Function 20-10
20.2.5 GET_HOUR Function 20-11
20.2.6 GET_DAY Function 20-11
20.2.7 GET_MONTH Function 20-12
20.2.8 GET_YEAR Function 20-13
20.2.9 IS_ALPHA Function 20-13
20.2.10 IS_DIGIT Function 20-14
20.2.11 IS_DVSYS_OWNER Function 20-15
20.2.12 IS_OLS_INSTALLED Function 20-15
20.2.13 IS_OLS_INSTALLED_VARCHAR Function 20-16
20.2.14 ROLE_GRANTED_ENABLED_VARCHAR Function 20-16
20.2.15 USER_HAS_OBJECT_PRIVILEGE Function 20-17
20.2.16 USER_HAS_ROLE Function 20-19
20.2.17 USER_HAS_ROLE_VARCHAR Function 20-19
20.2.18 USER_HAS_SYSTEM_PRIVILEGE Function 20-20

21 Oracle Database Vault General Administrative APIs

21.1 DBMS_MACADM General System Maintenance Procedures 21-1
21.1.1 ADD_APP_EXCEPTION Procedure 21-4
21.1.2 ADD_NLS_DATA Procedure 21-5
21.1.3 ALLOW_COMMON_OPERATION Procedure 21-6

21.1.4 AUTH_DATAPUMP_CREATE_USER Procedure 21-7
21.1.5 AUTH_DATAPUMP_GRANT Procedure 21-8
21.1.6 AUTH_DATAPUMP_GRANT_ROLE Procedure 21-8
21.1.7 AUTH_DATAPUMP_GRANT_SYSPRIV Procedure 21-9
21.1.8 AUTHORIZE_AUDIT_ADMIN Procedure 21-10
21.1.9 AUTHORIZE_AUDIT_VIEWER Procedure 21-10
21.1.10 AUTHORIZE_DATAPUMP_USER Procedure 21-11
21.1.11 AUTHORIZE_DBCAPTURE Procedure 21-12
21.1.12 AUTHORIZE_DBREPLAY Procedure 21-12
21.1.13 AUTHORIZE_DDL Procedure 21-13
21.1.14 AUTHORIZE_DIAGNOSTIC_ADMIN Procedure 21-14
21.1.15 AUTHORIZE_MAINTENANCE_USER Procedure 21-14
21.1.16 AUTHORIZE_PREPROCESSOR Procedure 21-15
21.1.17 AUTHORIZE_PROXY_USER Procedure 21-16
21.1.18 AUTHORIZE_SCHEDULER_USER Procedure 21-17
21.1.19 AUTHORIZE_SQL_FIREWALL Procedure 21-18
21.1.20 AUTHORIZE_TTS_USER Procedure 21-19
21.1.21 DELETE_APP_EXCEPTION Procedure 21-19
21.1.22 DISABLE_APP_PROTECTION Procedure 21-20
21.1.23 DISABLE_DV Procedure 21-21
21.1.24 DISABLE_DV_DICTIONARY_ACCTS Procedure 21-21
21.1.25 DISABLE_DV_PATCH_ADMIN_AUDIT Procedure 21-22
21.1.26 DISABLE_ORADEBUG Procedure 21-22
21.1.27 ENABLE_APP_PROTECTION Procedure 21-23
21.1.28 ENABLE_DV Procedure 21-23
21.1.29 ENABLE_DV_DICTIONARY_ACCTS Procedure 21-24
21.1.30 ENABLE_DV_PATCH_ADMIN_AUDIT Procedure 21-25
21.1.31 ENABLE_ORADEBUG Procedure 21-25
21.1.32 UNAUTH_DATAPUMP_CREATE_USER Procedure 21-26
21.1.33 UNAUTH_DATAPUMP_GRANT Procedure 21-26
21.1.34 UNAUTH_DATAPUMP_GRANT_ROLE Procedure 21-27
21.1.35 UNAUTH_DATAPUMP_GRANT_SYSPRIV Procedure 21-28
21.1.36 UNAUTHORIZE_ADMIN_USER Procedure 21-28
21.1.37 UNAUTHORIZE_AUDIT_VIEWER Procedure 21-29
21.1.38 UNAUTHORIZE_DATAPUMP_USER Procedure 21-29
21.1.39 UNAUTHORIZE_DBCAPTURE Procedure 21-30
21.1.40 UNAUTHORIZE_DBREPLAY Procedure 21-31
21.1.41 UNAUTHORIZE_DDL Procedure 21-31
21.1.42 UNAUTHORIZE_DIAGNOSTIC_ADMIN Procedure 21-32
21.1.43 UNAUTHORIZE_MAINTENANCE_USER Procedure 21-33
21.1.44 UNAUTHORIZE_PREPROCESSOR Procedure 21-34

21.1.45 UNAUTHORIZE_PROXY_USER Procedure 21-35
21.1.46 UNAUTHORIZE_SCHEDULER_USER Procedure 21-35
21.1.47 UNAUTHORIZE_SQL_FIREWALL Procedure 21-36
21.1.48 UNAUTHORIZE_TTS_USER Procedure 21-37
21.2 CONFIGURE_DV General System Maintenance Procedure 21-38

22 Oracle Database Vault Policy APIs

22.1 ADD_CMD_RULE_TO_POLICY Procedure 22-2
22.2 ADD_OWNER_TO_POLICY Procedure 22-3
22.3 ADD_REALM_TO_POLICY Procedure 22-4
22.4 CREATE_POLICY Procedure 22-5
22.5 DELETE_CMD_RULE_FROM_POLICY Procedure 22-6
22.6 DELETE_OWNER_FROM_POLICY Procedure 22-8
22.7 DELETE_REALM_FROM_POLICY Procedure 22-9
22.8 DROP_POLICY Procedure 22-9
22.9 RENAME_POLICY Procedure 22-10
22.10 UPDATE_POLICY_DESCRIPTION Procedure 22-10
22.11 UPDATE_POLICY_STATE Procedure 22-11

23 Oracle Database Vault API Reference

23.1 DBMS_MACADM PL/SQL Package Contents 23-1
23.2 DBMS_MACSEC_ROLES PL/SQL Package Contents 23-7
23.3 DBMS_MACUTL PL/SQL Package Contents 23-7
23.4 CONFIGURE_DV PL/SQL Procedure 23-8
23.5 DVF PL/SQL Interface Contents 23-8

24 Oracle Database Vault Data Dictionary Views

24.1 About the Oracle Database Vault Data Dictionary Views 24-5
24.2 CDB_DV_STATUS View 24-5
24.3 DBA_DV_APP_EXCEPTION View 24-6
24.4 DBA_DV_AUDIT_ADMIN_AUTH View 24-7
24.5 DBA_DV_AUDIT_VIEWER_AUTH View 24-7
24.6 DBA_DV_CODE View 24-8
24.7 DBA_DV_COMMAND_RULE View 24-9
24.8 DBA_DV_DATAPUMP_AUTH View 24-11
24.9 DBA_DV_DBCAPTURE_AUTH View 24-12
24.10 DBA_DV_DBREPLAY View 24-13
24.11 DBA_DV_DDL_AUTH View 24-13
24.12 DBA_DV_DICTIONARY_ACCTS View 24-14

24.13 DBA_DV_FACTOR View 24-14
24.14 DBA_DV_FACTOR_TYPE View 24-17
24.15 DBA_DV_FACTOR_LINK View 24-17
24.16 DBA_DV_IDENTITY View 24-18
24.17 DBA_DV_IDENTITY_MAP View 24-19
24.18 DBA_DV_JOB_AUTH View 24-20
24.19 DBA_DV_MAC_POLICY View 24-20
24.20 DBA_DV_MAC_POLICY_FACTOR View 24-21
24.21 DBA_DV_MAINTENANCE_AUTH View 24-21
24.22 DBA_DV_ORADEBUG View 24-22
24.23 DBA_DV_PATCH_ADMIN_AUDIT View 24-22
24.24 DBA_DV_POLICY View 24-23
24.25 DBA_DV_POLICY_LABEL View 24-24
24.26 DBA_DV_POLICY_OBJECT View 24-25
24.27 DBA_DV_POLICY_OWNER View 24-26
24.28 DBA_DV_PREPROCESSOR_AUTH View 24-27
24.29 DBA_DV_PROXY_AUTH View 24-27
24.30 DBA_DV_PUB_PRIVS View 24-28
24.31 DBA_DV_REALM View 24-29
24.32 DBA_DV_REALM_AUTH View 24-31
24.33 DBA_DV_REALM_OBJECT View 24-32
24.34 DBA_DV_ROLE View 24-33
24.35 DBA_DV_RULE View 24-34
24.36 DBA_DV_RULE_SET View 24-35
24.37 DBA_DV_RULE_SET_RULE View 24-38
24.38 DBA_DV_SIMULATION_LOG View 24-39
24.39 DBA_DV_STATUS or SYS.DBA_DV_STATUS View 24-42
24.40 DBA_DV_SQL_FIREWALL_AUTH View 24-43
24.41 DBA_DV_TTS_AUTH View 24-43
24.42 DBA_DV_USER_PRIVS View 24-44
24.43 DBA_DV_USER_PRIVS_ALL View 24-45
24.44 DVSYS.DV$CONFIGURATION_AUDIT View 24-46
24.45 DVSYS.DV$ENFORCEMENT_AUDIT View 24-50
24.46 DVSYS.DV$REALM View 24-53
24.47 DVSYS.DBA_DV_COMMON_OPERATION_STATUS View 24-55
24.48 DVSYS.POLICY_OWNER_COMMAND_RULE View 24-55
24.49 DVSYS.POLICY_OWNER_POLICY View 24-57
24.50 DVSYS.POLICY_OWNER_REALM View 24-57
24.51 DVSYS.POLICY_OWNER_REALM_AUTH View 24-59
24.52 DVSYS.POLICY_OWNER_REALM_OBJECT View 24-60
24.53 DVSYS.POLICY_OWNER_RULE View 24-61

24.54 DVSYS.POLICY_OWNER_RULE_SET View 24-62
24.55 DVSYS.POLICY_OWNER_RULE_SET_RULE View 24-64
24.56 AUDSYS.DV$CONFIGURATION_AUDIT View 24-65
24.57 AUDSYS.DV$ENFORCEMENT_AUDIT View 24-65

25 Monitoring Oracle Database Vault

25.1 About Monitoring Oracle Database Vault 25-1
25.2 Monitoring Security Violations and Configuration Changes 25-1

26 Oracle Database Vault Reports

26.1 About the Oracle Database Vault Reports 26-1
26.2 Who Can Run the Oracle Database Vault Reports? 26-2
26.3 Running the Oracle Database Vault Reports 26-2
26.4 Oracle Database Vault Configuration Issues Reports 26-2
26.4.1 Command Rule Configuration Issues Report 26-3
26.4.2 Rule Set Configuration Issues Report 26-3
26.4.3 Realm Authorization Configuration Issues Report 26-3
26.4.4 Factor Configuration Issues Report 26-4
26.4.5 Factor Without Identities Report 26-4
26.4.6 Identity Configuration Issues Report 26-4
26.4.7 Secure Application Configuration Issues Report 26-4
26.5 Oracle Database Vault Auditing Reports 26-5
26.5.1 Realm Audit Report 26-5
26.5.2 Command Rule Audit Report 26-5
26.5.3 Factor Audit Report 26-5
26.5.4 Label Security Integration Audit Report 26-6
26.5.5 Core Database Vault Audit Trail Report 26-6
26.5.6 Secure Application Role Audit Report 26-6
26.6 Oracle Database Vault General Security Reports 26-6
26.6.1 Object Privilege Reports 26-7
26.6.1.1 Object Access By PUBLIC Report 26-7
26.6.1.2 Object Access Not By PUBLIC Report 26-8
26.6.1.3 Direct Object Privileges Report 26-8
26.6.1.4 Object Dependencies Report 26-8
26.6.2 Database Account System Privileges Reports 26-9
26.6.2.1 Direct System Privileges By Database Account Report 26-9
26.6.2.2 Direct and Indirect System Privileges By Database Account Report 26-9
26.6.2.3 Hierarchical System Privileges by Database Account Report 26-9
26.6.2.4 ANY System Privileges for Database Accounts Report 26-10

26.6.2.5 System Privileges By Privilege Report 26-10
26.6.3 Sensitive Objects Reports 26-10
26.6.3.1 Execute Privileges to Strong SYS Packages Report 26-10
26.6.3.2 Access to Sensitive Objects Report 26-11
26.6.3.3 Public Execute Privilege To SYS PL/SQL Procedures Report 26-11
26.6.3.4 Accounts with SYSDBA/SYSOPER Privilege Report 26-12
26.6.4 Privilege Management - Summary Reports 26-12
26.6.4.1 Privileges Distribution By Grantee Report 26-12
26.6.4.2 Privileges Distribution By Grantee, Owner Report 26-12
26.6.4.3 Privileges Distribution By Grantee, Owner, Privilege Report 26-13
26.6.5 Powerful Database Accounts and Roles Reports 26-13
26.6.5.1 WITH ADMIN Privilege Grants Report 26-14
26.6.5.2 Accounts With DBA Roles Report 26-14
26.6.5.3 Security Policy Exemption Report 26-14
26.6.5.4 BECOME USER Report 26-14
26.6.5.5 ALTER SYSTEM or ALTER SESSION Report 26-14
26.6.5.6 Password History Access Report 26-15
26.6.5.7 WITH GRANT Privileges Report 26-15
26.6.5.8 Roles/Accounts That Have a Given Role Report 26-15
26.6.5.9 Database Accounts With Catalog Roles Report 26-15
26.6.5.10 AUDIT Privileges Report 26-16
26.6.5.11 OS Security Vulnerability Privileges Report 26-16
26.6.6 Initialization Parameters and Profiles Reports 26-16
26.6.6.1 Security Related Database Parameters Report 26-16
26.6.6.2 Resource Profiles Report 26-16
26.6.6.3 System Resource Limits Report 26-16
26.6.7 Database Account Password Reports 26-17
26.6.7.1 Database Account Default Password Report 26-17
26.6.7.2 Database Account Status Report 26-17
26.6.8 Security Audit Report: Core Database Audit Report 26-17
26.6.9 Other Security Vulnerability Reports 26-17
26.6.9.1 Java Policy Grants Report 26-18
26.6.9.2 OS Directory Objects Report 26-18
26.6.9.3 Objects Dependent on Dynamic SQL Report 26-18
26.6.9.4 Unwrapped PL/SQL Package Bodies Report 26-19
26.6.9.5 Username/Password Tables Report 26-19
26.6.9.6 Tablespace Quotas Report 26-19
26.6.9.7 Non-Owner Object Trigger Report 26-19

A Auditing Oracle Database Vault

A.1 About Auditing in Oracle Database Vault A-1
A.1.1 Auditing Oracle Database Vault Using Unified Auditing A-1
A.1.2 Auditing Oracle Database Vault Using Traditional Auditing A-2
A.2 Oracle Database Vault Specific Audit Events A-3
A.2.1 Oracle Database Vault Policy Audit Events A-3
A.2.2 Oracle Database Vault Audit Trail Record Format A-4
A.3 Archiving and Purging the Oracle Database Vault Audit Trail A-6
A.3.1 About Archiving and Purging the Oracle Database Vault Audit Trail A-6
A.3.2 Archiving the Oracle Database Vault Audit Trail A-6
A.3.3 Purging the Oracle Database Vault Audit Trail A-8
A.4 Oracle Database Vault Predefined Unified Audit Policies A-8

B Disabling and Enabling Oracle Database Vault

B.1 When You Must Disable Oracle Database Vault B-1
B.2 Step 1: Disable Oracle Database Vault B-2
B.3 Step 2: Perform the Required Tasks B-3
B.4 Step 3: Enable Oracle Database Vault B-3

C Postinstallation Oracle Database Vault Procedures

C.1 Adding Languages to Oracle Database Vault C-1
C.2 Uninstalling Oracle Database Vault C-2
C.3 Reinstalling Oracle Database Vault C-3

D Oracle Database Vault Security Guidelines

D.1 Separation of Duty Guidelines D-1
D.1.1 How Oracle Database Vault Handles Separation of Duty D-1
D.1.2 Separation of Tasks in an Oracle Database Vault Environment D-2
D.1.3 Separation of Duty Matrix for Oracle Database Vault D-3
D.1.4 Identification and Documentation of the Tasks of Database Users D-4
D.2 Managing Oracle Database Administrative Accounts D-5
D.2.1 SYSTEM User Account for General Administrative Uses D-5
D.2.2 SYSTEM Schema for Application Tables D-6
D.2.3 Limitation of the SYSDBA Administrative Privilege D-6
D.2.4 Root and Operating System Access to Oracle Database Vault D-6
D.3 Accounts and Roles Trusted by Oracle Database Vault D-7
D.4 Accounts and Roles That Should be Limited to Trusted Individuals D-8
D.4.1 Management of Users with Root Access to the Operating System D-8

D.4.2 Management of the Oracle Software Owner D-8
D.4.3 Management of SYSDBA Access D-9
D.4.4 Management of SYSOPER Access D-9
D.5 Guidelines for Using Oracle Database Vault in a Production Environment D-10
D.6 Secure Configuration Guidelines D-10
D.6.1 General Secure Configuration Guidelines D-10
D.6.2 UTL_FILE and DBMS_FILE_TRANSFER Package Security Considerations D-11
D.6.2.1 About Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages D-11
D.6.2.2 Securing Access to the DBMS_FILE_TRANSFER Package D-12
D.6.2.3 Example: Creating a Command Rule to Deny Access to CREATE DATABASE LINK D-12
D.6.2.4 Example: Creating a Command Rule to Enable Access to CREATE DATABASE LINK D-13
D.6.2.5 Example: Command Rules to Disable and Enable Access to CREATE DIRECTORY D-13
D.6.3 CREATE ANY JOB Privilege Security Considerations D-14
D.6.4 CREATE EXTERNAL JOB Privilege Security Considerations D-14
D.6.5 ALTER SYSTEM and ALTER SESSION Privilege Security Considerations D-14
D.6.5.1 About ALTER SYSTEM and ALTER SESSION Privilege Security Considerations D-14
D.6.5.2 Example: Adding Rules to the Existing ALTER SYSTEM Command Rule D-15

E Troubleshooting Oracle Database Vault

E.1 Using Trace Files to Diagnose Oracle Database Vault Events E-1
E.1.1 About Using Trace Files to Diagnose Oracle Database Vault Events E-2
E.1.2 Types of Oracle Database Vault Trace Events That You Can and Cannot Track E-2
E.1.3 Levels of Oracle Database Vault Trace Events E-3
E.1.4 Performance Effect of Enabling Oracle Database Vault Trace Files E-3
E.1.5 Enabling Oracle Database Vault Trace Events E-3
E.1.5.1 Enabling Trace Events for the Current Database Session E-3
E.1.5.2 Enabling Trace Events for All Database Sessions E-4
E.1.5.3 Enabling Trace Events in a Multitenant Environment E-5
E.1.6 Finding Oracle Database Vault Trace File Data E-5
E.1.6.1 Finding the Database Vault Trace File Directory Location E-6
E.1.6.2 Using the Linux grep Command to Search Trace Files for Strings E-6
E.1.6.3 Using the ADR Command Interpreter (ADRCI) Utility to Query Trace Files E-6
E.1.6.4 Example: Low Level Oracle Database Vault Realm Violations in a Trace File E-7
E.1.6.5 Example: High Level Trace Enabled for Oracle Database Vault Authorization E-8

E.1.6.6 Example: Highest Level Traces on Violations on Realm-Protected Objects E-9
E.1.7 Disabling Oracle Database Vault Trace Events E-10
E.1.7.1 Disabling Trace Events for the Current Database Session E-10
E.1.7.2 Disabling Trace Events for All Database Sessions E-10
E.1.7.3 Disabling Trace Events in a Multitenant Environment E-11
E.2 General Diagnostic Tips E-11
E.3 Configuration Problems with Oracle Database Vault Components E-11
E.4 Resetting Oracle Database Vault Account Passwords E-12
E.4.1 Resetting the DV_OWNER User Password E-12
E.4.2 Resetting the DV_ACCTMGR User Password E-13

Index

Preface

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

- Audience
- Documentation Accessibility
- Diversity and Inclusion
- Related Documents
- Conventions

Audience

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

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

For more information refer to the following documents:

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

Oracle Technical Services

To download the product data sheet, frequently asked questions, links to the latest product documentation, product download, and other collateral, visit Oracle Technical Resources (formerly Oracle Technology Network). You must register online before using Oracle Technical Services. Registration is free and can be done at

`https://www.oracle.com/technical-resources/`

My Oracle Support

You can find information about security patches, certifications, and the support knowledge base by visiting My Oracle Support (formerly Oracle*MetaLink*) at

`https://support.oracle.com`

Conventions

The following text conventions are used in this document:

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

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

This preface contains:

- Changes in Oracle Database Vault 23c

Changes in Oracle Database Vault 23c

The following are changes in *Oracle Database Vault Administrator's Guide* for Oracle Database 23c.

- Ability to Control Authorizations for Unified Auditing and Traditional Auditing
  Starting with Oracle Database 23c, you can have tighter control on audit management with audit authorizations.
- Ability to Control Authorizations for SQL Firewall
  Starting with Oracle Database 23c, you can control the authorization of users to use SQL Firewall, new to this release, in an Oracle Database Vault environment.
- Desupport of Traditional Auditing in Oracle Database Vault
  Starting with Oracle Database 23c, traditional auditing is desupported.

Ability to Control Authorizations for Unified Auditing and Traditional Auditing

Starting with Oracle Database 23c, you can have tighter control on audit management with audit authorizations.

Starting in this release, when Oracle Database Vault is configured and enabled on a new or upgraded, database, audit authorizations will be required for any user who wants to query audit-related tables and views, or manage traditional or unified audit policies, without first being authorized by Oracle Database Vault. This applies to `SYS`, `SYSTEM`, and any user with the `SYSDBA` administrative privilege, or with the `DBA`, `AUDIT_ADMIN`, or `AUDIT_VIEWER` roles.

This feature provides the following audit-related enhancements for an Oracle Database Vault environment:

- It protects both unified auditing and traditional auditing.
- It blocks direct modification of the `SYS.AUD$` and `SYS.FGA_LOG$` database tables except through the `DBMS_AUDIT_MGMT` PL/SQL package by authorized users.
- It provides a new mandatory default realm for audit-related objects, `Oracle Audit Realm`, to protect the `AUDSYS` schema and audit-related objects in the `SYS` schema.

The following new procedures are available for this feature:

- DVSYS.DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN
- DVSYS.DBMS_MACADM.UNAUTHORIZE_AUDIT_ADMIN
- DVSYS.DBMS_MACADM.AUTHORIZE_AUDIT_VIEWER
- DVSYS.DBMS_MACADM.UNAUTHORIZE_AUDIT_VIEWER

The following new data dictionary views are available:

- DVSYS.DBA_DV_AUDIT_ADMIN_AUTH
- DVSYS.DBA_DV_AUDIT_VIEWER_AUTH

The ALTER SYSTEM command rule now requires AUDIT_ADMIN authorization for the AUDIT_FILE_DEST, AUDIT_TRAIL, AUDIT_SYS_OPERATIONS, and AUDIT_SYSLOG_LEVEL parameters. (Note that these parameteres have been deprecated starting in this release.) Querying audit trails in the SYS and AUDSYS schemas now requires AUDIT_ADMIN or AUDIT_VIEWER authorization.

In previous releases, to control or restrict auditing, the Database Vault administrator had to create command rules for the audit-related PL/SQL statements such as CREATE AUDIT POLICY. Traditional auditing requires the modification of system parameters such as AUDIT_FILE_DEST. This new authorization does not consolidate or replace the required database privileges. The audit authorization is additional requirement for managing audit in an Oracle Database Vault environment. That is, users are required to have sufficient privileges *and* audit authorization in order to manage audit when Oracle Database Vault is enabled. In addition to facilitating the granting of audit-related privileges to the user, this enhancement provides greater separation of duties for managing auditing in an Oracle Database Vault environment.

Related Topics

- Using Oracle Database Auditing with Oracle Database Vault
  You grant and revoke authorizations to manage and view audit records in an Oracle Database Vault environment.

Ability to Control Authorizations for SQL Firewall

Starting with Oracle Database 23c, you can control the authorization of users to use SQL Firewall, new to this release, in an Oracle Database Vault environment.

In addition to granting a user the ability to perform SQL Firewall operations, you can prevent the user from using SQL Firewall to apply policies that could affect users who have been granted the DV_OWNER and DV_ACCTMGR roles.

The following new DBMS_MACADM package procedures are available for this feature:

- DVSYS.DBMS_MACADM.AUTHORIZE_SQL_FIREWALL
- DVSYS.DBMS_MACADM.UNAUTHORIZE_SQL_FIREWALL

This enhancement also includes the DBA_DV_SQL_FIREWALL_AUTH data dictionary view, which provides information about SQL Firewall authorizations.

Related Topics

- Using SQL Firewall with Oracle Database Vault
  You can authorize SQL Firewall users to work in a Database Vault environment.

Desupport of Traditional Auditing in Oracle Database Vault

Starting with Oracle Database 23c, traditional auditing is desupported.

Unified auditing is the way forward to perform Oracle Database Vault auditing. Unified auditing offers more flexibility to perform selective and effective auditing, which helps you focus on activities that really matter to your enterprise. Unified auditing has one single and secure unified trail, conditional policy for audit selectivity, and default predefined policies for simplicity. To improve security and compliance, Oracle strongly recommends that you use unified auditing.

The main impact of the desupport of traditional auditing in Oracle Database Vault is with the `audit_options` parameter in the APIs for realms, rule sets, and factors.

Related Topics

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

1

Introduction to Oracle Database Vault

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

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

1.1 What Is Oracle Database Vault?

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

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

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

1.1.1 About Oracle Database Vault

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

Oracle Database Vault is a licensable option of Oracle Database Enterprise Edition. Its purpose is to mitigate the potential impact of privileged account abuse, misuse, insider and external threats, and human error on your sensitive data.

Privileged accounts can be administrator accounts, such as database administrators, or application administrators, application owners, or data analysts. Most users with these type of accounts have far more privileges and access than they need on a daily basis.

Oracle Database Vault is built into the kernel of the Oracle database and makes decisions after system or object privileges are verified. If the command is authorized by a system or object privilege, then Oracle Database Vault then determines if the command is controlled by an Oracle Database Vault realm or command rule. The realm or command rule controls are determined by you. Oracle Database Vault does not replace your existing existing grants or roles but augments them by allowing you to decide when, where, why, and how object or system privileges or roles are used by the grantee.

For example, any user who is granted the `SELECT ANY TABLE` system privilege can use their privilege to query virtually any table in the database, including tables you consider sensitive or important. Oracle Database Vault can restrict the `SELECT ANY TABLE` system privilege usage on your sensitive tables or other database objects.

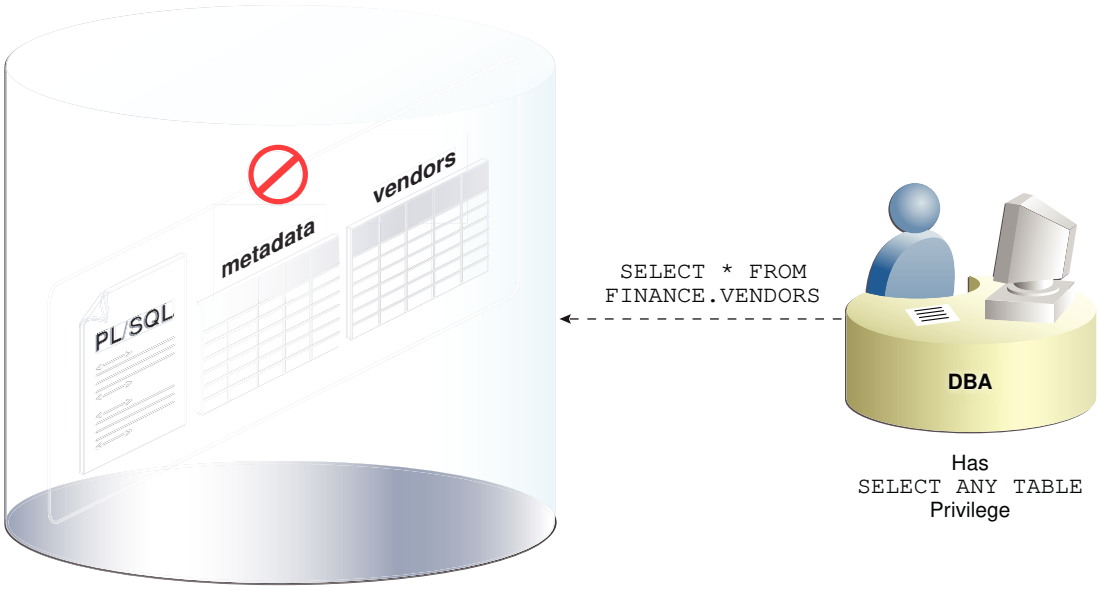
Oracle Database Vault helps you increase the security of existing applications through transparent controls that are determined by your requirements.

1.1.2 Controls for Privileged Accounts

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

While their broad and unrestricted access facilitates database maintenance, the same access also creates a point of attack for gaining access to large amounts of data. Oracle Database Vault realms around application schemas, sensitive tables, and stored procedures provide controls to prevent privileged accounts from being exploited by intruders and insiders to access sensitive application data.

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



1.1.3 Controls for Database Configuration

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

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

The Oracle Database Vault realms with the mandatory mode enables you to seal off access to application objects, even to those with direct object grants, including the object owner. With mandatory realms, you do not need to analyze who has access because this is clear from the list of authorized users.

1.1.4 Enterprise Applications Protection Policies

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

These enterprise applications include Oracle Fusion Applications, Oracle E-Business Suit, Oracle PeopleSoft, Oracle Siebel, Oracle Financial Services (i-Flex), Oracle Primavera, SAP, and Finacle from Infosys. Because Oracle Database Vault does not modify the application nor require changes to the client, you can use it with most off-the-shelf and custom applications.

1.2 What Privileges Do You Need to Use Oracle Database Vault?

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

The most commonly used roles are as follows:

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

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

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

Related Topics

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

1.3 Components of Oracle Database Vault

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

- Oracle Database Vault Access Control Components
  Oracle Database Vault enables you to create a set of components to manage security for your database instance.
- Oracle Database Vault DVSYS and DVF Schemas
  Oracle Database Vault database objects and public functions are stored in the `DVSYS` and `DVF` schemas, respectively.
- Oracle Database Vault PL/SQL Interfaces and Packages
  Oracle Database Vault provides PL/SQL interfaces and packages for security managers or application developers to configure access control policies.
- Oracle Database Vault Reporting and Monitoring Tools
  Oracle Enterprise Manager generates and maintains the Oracle Database Vault reports.

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

1.3.1 Oracle Database Vault Access Control Components

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

These components are as follows:

- **Realms.** A realm is a protection zone inside the database where database schemas, objects, and roles can be secured. For example, you can secure a set of schemas, objects, and roles that are related to accounting, sales, or human resources. After you have secured these into a realm, you can use the realm to control the use of system and object privileges to specific accounts or roles. This enables you to provide fine-grained access controls for anyone who wants to use these schemas, objects, and roles Configuring Realms, discusses realms in detail. See also Oracle Database Vault Realm APIs.
- **Command rules.** A command rule is a special security policy that you can create to control how users can run almost any SQL statement, including `SELECT`, `ALTER SYSTEM`, database definition language (DDL), and data manipulation language (DML) statements. Command rules use rule sets to determine whether the statement is allowed. Configuring Command Rules , discusses command rules in detail. See also Oracle Database Vault Command Rule APIs.
- **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**). Rule sets can be associated with zero, one, or multiple realm authorizations, command rules, or secure application roles. Configuring Rule Sets , discusses rule sets in detail. See also Oracle Database Vault Rule Set APIs.
- **Rules.** A rule is a PL/SQL expression that evaluates to true or false. You can use the same rule in multiple rule sets. For more information, see How Rule Sets Work.
- **Factors.** A factor is a named variable or attribute, such as a user location, database IP address, or session user, which Oracle Database Vault can recognize and use as a trusted path. You can use factors in rules to control activities such as authorizing database accounts to connect to the database or the execution of a specific database command to restrict the visibility and manageability of data. Each factor can have one or more identities. An identity is the actual value of a factor. A factor can have several identities depending on the factor retrieval method or its identity mapping logic. Configuring Factors, discusses factors in detail. See also Oracle Database Vault Factor APIs.
- **Secure application roles.** A secure application role is a special Oracle Database role that can be enabled based on the evaluation of an Oracle Database Vault rule set. Configuring Secure Application Roles for Oracle Database Vault, discusses secure application roles in detail. See also Oracle Database Vault Secure Application Role APIs .

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

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

1.3.2 Oracle Database Vault DVSYS and DVF Schemas

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

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

Related Topics

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

1.3.3 Oracle Database Vault PL/SQL Interfaces and Packages

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

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

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

1.3.4 Oracle Database Vault Reporting and Monitoring Tools

Oracle Enterprise Manager generates and maintains the Oracle Database Vault reports.

Oracle Database Vault provides database views that enable you to retrieve information about Oracle Database Vault configuration settings, including status and component information.

In addition, you can monitor policy changes, security violation attempts, and Oracle Database Vault configuration and structure changes through the Oracle Database unified audit trail or Oracle Enterprise Manager.

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.
- Monitoring Oracle Database Vault
  You can monitor Oracle Database Vault by checking for violations to the Database Vault configurations and by tracking changes to policies.

1.3.5 Oracle Enterprise Manager Cloud Control Database Vault Administrator Pages

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

Oracle Enterprise Manager Cloud Control provides a graphical user interface you can use to view and configure Oracle Database Vault policies and view Oracle Database Vault alerts and reports. Oracle Database Vault Administrator provides an extensive collection of security-related reports that assist in understanding the baseline security configuration.

1.4 How Oracle Database Vault Addresses Compliance Regulations

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

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

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

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

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

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

Table 1-1 Regulations That Address Potential Security Threats

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

1.5 How Oracle Database Vault Protects Privileged User Accounts

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

Oracle Database Vault helps to protect against compromised privilege user account attacks by using realms, factors, and command rules. Combined, these provide powerful security tools to help secure access to databases, applications, and sensitive information. You can combine rules and factors to control the conditions under which commands in the database are allowed to run, 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 program, such as JDBC, SQL Developer, or SQL*Plus. These can limit access to only those connections that pass these conditions. This can prevent unauthorized access to application data and access to the database by unauthorized applications. For example, you could define a rule to limit execution of the `DROP TABLE` statement to a specific IP address and host name.

1.6 How Oracle Database Vault Allows for Flexible Security Policies

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

For example, any database user who has the `DBA` role can use the `DROP ANY TABLE` system privilege granted to that role. Suppose an inexperienced administrator believes they are on a non-production database when they execute a `DROP TABLE` command and is instead on the production system and drops a critical application table. This will probably cause an application outage, data loss, and hours to recover from. With Oracle Database Vault, you can create a command rule to prevent this user from

making such modifications by limiting their usage of the `DROP TABLE` statement. Furthermore, you can attach rule sets to the command rule to restrict activity further, such as limiting the statement's execution in the following ways:

- By time (for example, only outside of business hours of 8 a.m. to 6 p.m., Monday through Friday)
- By local access only, that is, not remotely
- Require two database users to authorize an action instead of one user
- If the user has an Oracle Database Vault secure application role enabled
- By host name or IP address (for example, the host name could be `%appserver%` or match an IP address of 192.0.2.150

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

1.7 How Oracle Database Vault Addresses Database Consolidation Concerns

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

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

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

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

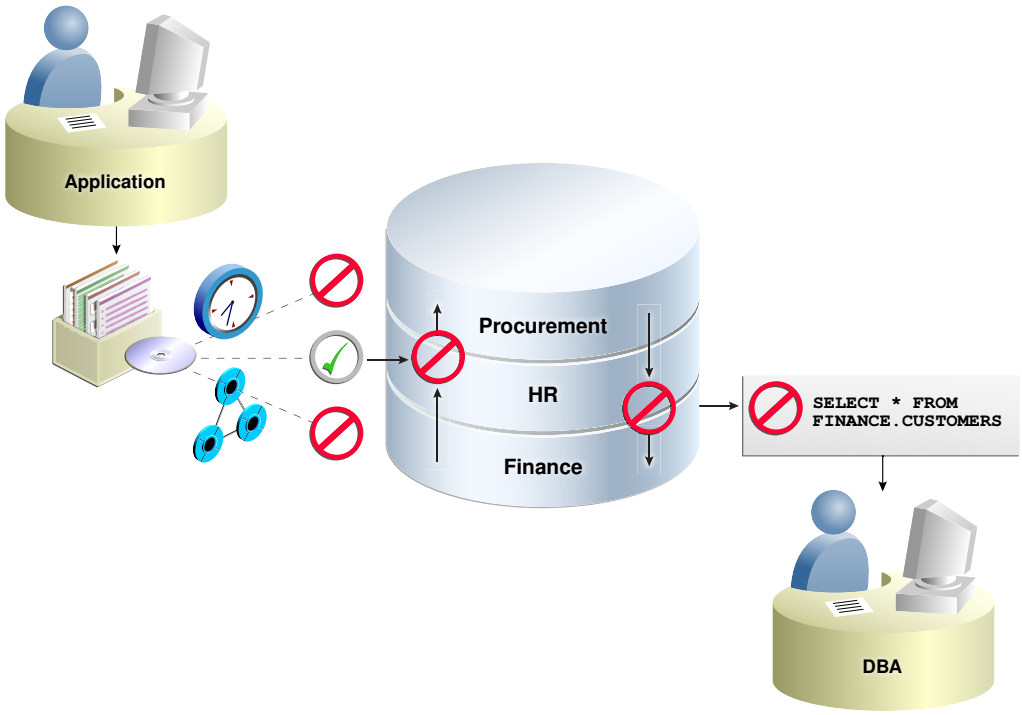
Oracle Database Vault separation of duty controls can be customized and organizations with limited resources can assign multiple Oracle Database Vault responsibilities to the same administrator, but using separate accounts for each separation-of-duty role to minimize damage to the database if any one account is stolen and leveraged.

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

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

- **Administrative privileged account access to application data**: In this case, Oracle Database Vault prevents the database administrator from accessing the schemas that are protected by the Finance realm. Although the database administrator is the most powerful and trusted user, this administrator does not need access to application data residing within the database.
- **Separation of duties for application data access**: In this case, the `HR` realm owner has access to the `HR` realm schemas but not `Procurement` or `Finance`.

Figure 1-2 Oracle Database Vault Security



Database consolidation can result in multiple powerful user accounts residing in a single database. This means that in addition to the overall database administrator, individual application schema owners also may have powerful privileges. Revoking some privileges may adversely affect existing applications. Using Oracle Database Vault realms, you can enforce access to applications through a trusted application path, preventing the application schema user name and password from being used by anyone other than the application itself. For example, a database administrator who has the `SELECT ANY TABLE` system privilege can be prevented from using that privilege to view other application data residing in the same database.

1.8 How Oracle Database Vault Works in a Multitenant Environment

Using Oracle Database Vault in a multitenant environment increases security for consolidation.

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

You can configure common realms in the application root only, but you can create common rule sets and command rules in either the application root or the CDB root. A common command rule in the application root applies to its associated PDBs, and common command rules in the CDB root apply to all PDBs in the CDB environment. The ability to create common realms and command rules enables you to create policies that use a shared set of realms, rule sets, or command rules throughout the CDB environments, rather than having to create these same components for every PDB in the multitenant environment.

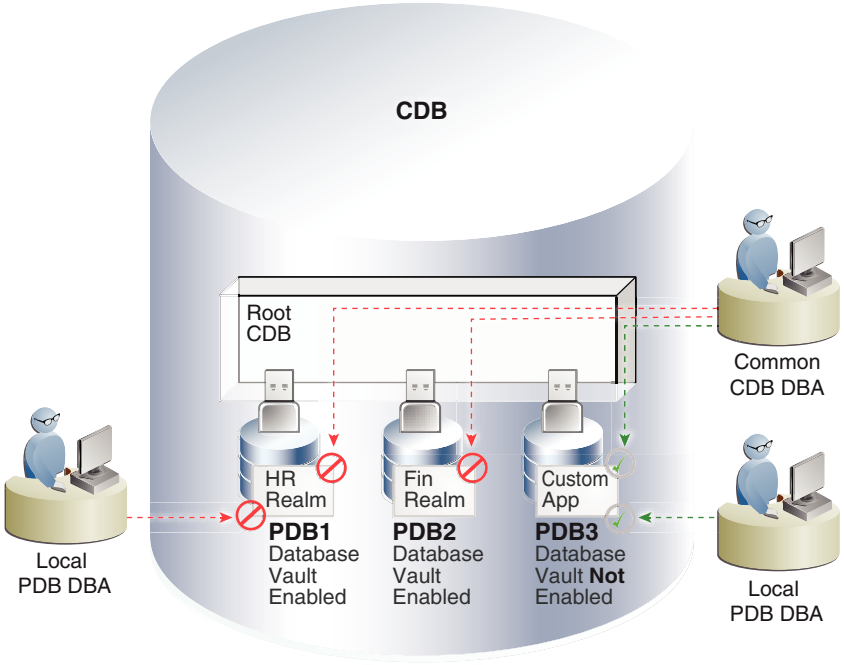
You can create individual local policies for each PDB. When you use Database Vault to protect an object, Database Vault subjects common privileges for common objects to the same enforcement rules as local system privileges.

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

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

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

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



Related Topics

- Realms in a Multitenant Environment
  You can create a realm to protect common objects in the application root.
- Rule Sets and Rules in a Multitenant Environment
  You can create a rule set and its associated rules in a PDB or an application root.
- Command Rules in a Multitenant Environment
  You can create common and local command rules in either the CDB root or the application root.
- Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB
  You can convert a standalone Oracle Database database from release 12c through 19c to a PDB, and then plug this PDB into a CDB.

2

What to Expect After You Enable Oracle Database Vault

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

- Initialization and Password Parameter Settings That Change
  The Oracle Database Vault configuration modifies several database initialization parameter settings to better secure your database configuration.
- How Oracle Database Vault Restricts User Authorizations
  The Oracle Database configuration requires four administrative database account names (two primary accounts and two backup accounts).
- Oracle Database Vault-Specific Database Roles to Enforce Separation of Duties
  The Oracle Database Vault configuration implements the concept of *separation of duty* so that you can improve security and meet regulatory, privacy, and other compliance requirements.
- Privileges That Are Revoked from Existing Users and Roles
  The Oracle Database Vault configuration revokes privileges from several Oracle Database-supplied users and roles, for better separation of duty.
- Privileges That Are Prevented for Existing Users and Roles
  The Oracle Database Vault configuration prevents several privileges for all users and roles who have been granted these privileges, including users `SYS` and `SYSTEM`.

2.1 Initialization and Password Parameter Settings That Change

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

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

Table 2-1 describes the initialization parameter settings that Oracle Database Vault modifies. Initialization parameters are stored in the `init.ora` initialization parameter file. See *Oracle Database Reference* for more information about initialization parameters.

Table 2-1 Modified Database Initialization Parameter Settings

Parameter	Default Value in Database	New Value Set by Database Vault	Impact of the Change
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.
REMOTE_LOGIN_PASSWORDFILE	EXCLUSIVE	EXCLUSIVE	Specifies whether Oracle Database checks for a password file. The EXCLUSIVE setting enforces the use of the password file, if you installed Oracle Database Vault into a database where REMOTE_LOGIN_PASSWORDFILE is not set to EXCLUSIVE.
SQL92_SECURITY	TRUE	TRUE	Ensures that if a user has been granted the UPDATE or DELETE object privilege, then the user must also be granted the SELECT object privilege before being able to perform UPDATE or DELETE operations on tables that have WHERE or SET clauses. Be aware that if the user is only granted the READ object privilege (instead of SELECT), then the user is not able to perform UPDATE or DELETE operations.

2.2 How Oracle Database Vault Restricts User Authorizations

The Oracle Database configuration requires four administrative database account names (two primary accounts and two backup accounts).

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

Related Topics

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

2.3 Oracle Database Vault-Specific Database Roles to Enforce Separation of Duties

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

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

Related Topics

- Separation of Duty Guidelines
  Oracle Database Vault is designed to easily implement separation of duty guidelines.
- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.

2.4 Privileges That Are Revoked from Existing Users and Roles

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

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

Table 2-2 Privileges Oracle Database Vault Revokes

User or Role	Privilege That Is Revoked
DBA role	• BECOME USER • SELECT ANY TRANSACTION • CREATE ANY JOB • CREATE EXTERNAL JOB • EXECUTE ANY PROGRAM • EXECUTE ANY CLASS • MANAGE SCHEDULER • DEQUEUE ANY QUEUE • ENQUEUE ANY QUEUE • MANAGE ANY QUEUE
IMP_FULL_DATABASE role	• BECOME USER • MANAGE ANY QUEUE
EXECUTE_CATALOG_ROLE role	• EXECUTE ON DBMS_LOGMNR_D • EXECUTE ON DBMS_LOGMNR_LOGREP_DICT • EXECUTE ON DBMS_FILE_TRANSFER • EXECUTE ON SYS.DBMS_LOGMNR
PUBLIC user	• EXECUTE ON UTL_FILE during the execution of the CONFIGURE_DV procedure, but before this revocation takes place, CONFIGURE_DV grants the object privilege directly to any schema that is dependent on this procedure
SCHEDULER_ADMIN role	• CREATE ANY JOB • CREATE EXTERNAL JOB • EXECUTE ANY PROGRAM • EXECUTE ANY CLASS • MANAGE SCHEDULER

Note:

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

Related Topics

- Privileges of Oracle Database Vault Roles
  The Oracle Database Vault roles are designed to provide the maximum benefits of separation of duty.
- DV_ACCTMGR Database Vault Account Manager Role
  The DV_ACCTMGR role is a powerful role, used for accounts management.

2.5 Privileges That Are Prevented for Existing Users and Roles

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

The `DV_ACCTMGR` role has these privileges for separation of duty:

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

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

Any role can be granted to user `SYS`, but `SYS` cannot use the role because no roles are enabled in the `SYS` session.

3
Getting Started with Oracle Database Vault

Before you can start using Oracle Database Vault, you must configure and enable it with the Oracle database.

- About Configuring and Enabling Oracle Database Vault in Oracle Database
  Oracle Database includes Database Vault when you choose to include a default database in the installation process, but you must configure and enable it before you can use it.
- Configuring and Enabling Oracle Database Vault
  You can configure and enable Oracle Database Vault based on several scenarios.
- Verifying That Database Vault Is Configured and Enabled
  The `DBA_DV_STATUS`, `CDB_DV_STATUS`, and `DBA_OLS_STATUS` data dictionary views verify if Oracle Database is configured and enabled.
- Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control
  Oracle Enterprise Manager Cloud Control (Cloud Control) provides pages for managing Oracle Database Vault.
- Quick Start Tutorial: Securing a Schema from DBA Access
  This tutorial shows how to create a realm around the `HR` schema.

3.1 About Configuring and Enabling Oracle Database Vault in Oracle Database

Oracle Database includes Database Vault when you choose to include a default database in the installation process, but you must configure and enable it before you can use it.

The configuration and enablement process enables Oracle Label Security if it is not already enabled. Oracle Label Security is required for Oracle Database Vault but it does not require a separate license unless you begin using Oracle Label Security separately and create Oracle Label Security policies.

If you create a custom database, then you can use DBCA to install and enable Database Vault for it. The registration process enables Oracle Label Security if it is not already enabled. This procedure applies to the CDB root, application root, and the current pluggable database (PDB), as well as to both single-instance and Oracle Real Application Clusters (Oracle RAC) installations. In a multitenant database, Database Vault must be configured with the CDB root before any of the PDBs can configure Database Vault.

As part of the configuration process, you created the Oracle Database Vault administrative accounts. The Oracle Database Vault configuration recommends four administrative database account names (two primary accounts and two backup accounts) with different passwords for each. These are accounts that hold the Database Vault roles `DV_OWNER` and `DV_ACCTMGR`, granted with the `WITH ADMIN OPTION` clause. Two of these accounts will be used to provision the roles to named users with administrative privileges. Maintaining two backup accounts, one with `DV_OWNER` and one with `DV_ACCTMGR` roles, will allow you to recover from the named user losing or somehow misplacing their credentials because `SYS`, or any other user, will not be able to reset these passwords for users with these roles.

> **Note:**
>
> If you upgraded from a release earlier than Oracle Database 12c, then you disabled Oracle Database Vault to perform the upgrade. After the upgrade process is complete, then you must configure and enable Oracle Database Vault again.
> If you are migrating a non-Database Vault-enabled Oracle database from a release earlier than release 12c, then you must perform a manual installation of Database Vault.

Related Topics

- Manually Installing Oracle Database Vault
  Under certain conditions, you must manually install Oracle Database Vault.
- Verifying That Database Vault Is Configured and Enabled
  The `DBA_DV_STATUS`, `CDB_DV_STATUS`, and `DBA_OLS_STATUS` data dictionary views verify if Oracle Database is configured and enabled.
- Configuring and Enabling Oracle Database Vault
  You can configure and enable Oracle Database Vault based on several scenarios.

3.2 Configuring and Enabling Oracle Database Vault

You can configure and enable Oracle Database Vault based on several scenarios.

- About Configuring and Enabling Database Vault
  You must configure and enable Oracle Database Vault in the CDB root before you can perform the same action in any of the associated PDBs.
- Configuring and Enabling Database Vault in the CDB Root
  You can configure and enable Oracle Database Vault with common users who will use the Database Vault-enforced roles in the CDB root.
- Configuring and Enabling Database Vault Common Users to Manage Specific PDBs
  You must configure and enable Oracle Database Vault in the root first, then in the PDBs afterward.
- Configuring and Enabling Database Vault Local Users to Manage Specific PDBs
  You must configure and enable Oracle Database Vault in the root first, and then in the PDBs afterward.
- Configuring and Enabling Oracle Database Vault in an Oracle Real Application Clusters Environment
  You can configure Oracle Database Vault for an Oracle Real Application Clusters (Oracle RAC) environment, including each Oracle RAC node.
- Creating a Profile to Protect the DV_OWNER and DV_ACCTMGR Users
  A profile provides additional protection for users who have been granted the `DV_OWNER` and `DV_ACCTMGR` roles.
- Manually Installing Oracle Database Vault
  Under certain conditions, you must manually install Oracle Database Vault.

3.2.1 About Configuring and Enabling Database Vault

You must configure and enable Oracle Database Vault in the CDB root before you can perform the same action in any of the associated PDBs.

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

3.2.2 Configuring and Enabling Database Vault in the CDB Root

You can configure and enable Oracle Database Vault with common users who will use the Database Vault-enforced roles in the CDB root.

Before you begin, Oracle recommends that you ensure that all database-related objects are valid. You can use the `UTL_RECOMP` PL/SQL package to check the validity of objects. See *Oracle Database PL/SQL Packages and Types Reference*.

1. Log into the root of the database instance as a user who has privileges to create users and grant the `CREATE SESSION` and `SET CONTAINER` privileges.
2. Select user accounts (or create new users) that will be used for the Database Vault Owner (`DV_OWNER` role) and Database Vault Account Manager (`DV_ACCTMGR` role) accounts.

   Oracle strongly recommends that you maintain two accounts for each role. One account, the primary named user account, will be used on a day-to-day basis and the other account will be used as a backup account in case the password of the primary account is lost and must be reset. If you do not have a backup for your account, then you cannot reset passwords. In addition, you must grant the `DV_OWNER` role, `WITH ADMIN OPTION`, to be able to reset the password of the primary account. Store these passwords in a safe location, such as a privileged account management (PAM) system, in case they are needed in the future.

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

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

   This specification grants two system privileges, creates the accounts if they do not exist, assigns a password, and does this so all the users have access to the CDB and all PDB databases.

   - Create the primary accounts (`c##sec_admin_owen` and `c##accts_admin_ace`) if these do not already exist for the new roles, `DV_ADMIN` and `DV_ACCTMGR`.

- Replace *password* with a password that meets the password complexity requirements of the user's profile.

3. Connect to the root as user SYS with the SYSDBA administrative privilege
4. Configure the two backup Database Vault user accounts.

   For example:

```
BEGIN
 CONFIGURE_DV (
   dvowner_uname         => 'c##dbv_owner_root_backup',
   dvacctmgr_uname       => 'c##dbv_acctmgr_root_backup',
   force_local_dvowner   => FALSE);
 END;
/
```

   In this example, setting force_local_dvowner to FALSE enables the common users to have DV_OWNER privileges for the PDBs that are associated with this CDB root. Setting it to TRUE restricts the common DV_OWNER user to have the DV_OWNER role privileges for the CDB root only. If you grant DV_OWNER locally to the CDB root common user, then that user cannot grant the DV_OWNER role commonly to any other user.

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

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

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

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

   For example:

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

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

   - To enable Oracle Database Vault to use regular mode:

```
EXEC DBMS_MACADM.ENABLE_DV;
```

   - If every associated PDB will need to have Database Vault enabled in this database, then use the following command. (You will need to enable each of these PDBs after you complete this procedure.) PDBs that do not have Database Vault enabled will be in restricted mode after the database is restarted and until Database Vault is enabled in the PDB:

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

8. Connect with the SYSOPER administrative privilege.
9. Restart the database.

For a single-instance database:

```
SHUTDOWN IMMEDIATE
STARTUP
```

If you are in an Oracle Real Application Clusters (Oracle RAC) environment, then you can perform an Oracle RAC rolling enablement.

10. Connect with the `SYSDBA` administrative privilege.
11. Verify that Oracle Database Vault and Oracle Label Security are installed and enabled.

```
SELECT * FROM CDB_DV_STATUS;
SELECT * FROM CDB_OLS_STATUS;
```

12. Connect as the backup `DV_OWNER` user and then grant the `DV_OWNER` role, including the `WITH ADMIN OPTION` clause, to the primary `DV_OWNER` user that you created earlier.

    For example:

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

GRANT DV_OWNER TO c##sec_admin_owen WITH ADMIN OPTION CONTAINER = ALL;
```

13. Connect as the backup `DV_ACCTMGR` user and then grant the `DV_ACCTMGR` role, including the `WITH ADMIN OPTION` clause, to the backup `DV_ACCTMGR` user.

    For example:

```
CONNECT c##dbv_acctmgr_root_backup
Enter password: password

GRANT DV_ACCTMGR TO c##accts_admin_ace WITH ADMIN OPTION CONTAINER=ALL;
```

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

Related Topics

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

Related Topics

- DV_PATCH_ADMIN Database Vault Database Patch Role
  The `DV_PATCH_ADMIN` role is used for patching operations.

- CONFIGURE_DV General System Maintenance Procedure
  The `CONFIGURE_DV` procedure configures the initial two Oracle Database user accounts, which are granted the `DV_OWNER` and `DV_ACCTMGR` roles, respectively.
- Configuring and Enabling Oracle Database Vault in an Oracle Real Application Clusters Environment
  You can configure Oracle Database Vault for an Oracle Real Application Clusters (Oracle RAC) environment, including each Oracle RAC node.
- Resetting Oracle Database Vault Account Passwords
  Backup accounts can help you reset lost passwords for users who have been granted the `DV_OWNER` and `DV_ACCTMGR` roles.

3.2.3 Configuring and Enabling Database Vault Common Users to Manage Specific PDBs

You must configure and enable Oracle Database Vault in the root first, then in the PDBs afterward.

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

1. If you have not already done so, then identify or create named common user accounts to be used as the Database Vault accounts along with associated backup accounts.
2. Ensure that you have configured and enabled Oracle Database Vault in the CDB root and that the `DV_OWNER` role was granted commonly to the common user.
3. Connect to the PDB as an administrator who is local to the PDB.
4. If necessary, open the database.

   ```
   ALTER DATABASE OPEN;
   ```

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

   For example:

   ```
   GRANT CREATE SESSION, SET CONTAINER TO c##sec_admin_owen CONTAINER
   = CURRENT;
   GRANT CREATE SESSION, SET CONTAINER TO c##accts_admin_ace CONTAINER
   = CURRENT;
   ```

6. Connect as user `SYS` with the `SYSDBA` administrative privilege
7. While still in the PDB, configure the two backup Database Vault user accounts.

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

In this example, the `force_local_dvowner` parameter is omitted because it is unnecessary. All common users who are configured within a PDB are restricted to the scope of the PDB.

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

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

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

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

   For example:

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

10. Enable Oracle Database Vault in this PDB.

    ```
    EXEC DBMS_MACADM.ENABLE_DV;
    ```

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

12. Close and reopen the PDB.

    For example:

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

13. Verify that the PDB is configured and enabled for Database Vault and Oracle Label Security.

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

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

14. Connect as the backup `DV_OWNER` user and then grant the `DV_OWNER` role, including the `WITH ADMIN OPTION` clause, to the primary `DV_OWNER` user that you created earlier.

    For example:

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

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

15. Connect as the backup `DV_ACCTMGR` user and then grant the `DV_ACCTMGR` role, including the `WITH ADMIN OPTION` clause, to the primary `DV_ACCTMGR` user.

For example:

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

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

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

Related Topics

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

Related Topics

- DV_PATCH_ADMIN Database Vault Database Patch Role
  The `DV_PATCH_ADMIN` role is used for patching operations.
- CONFIGURE_DV General System Maintenance Procedure
  The `CONFIGURE_DV` procedure configures the initial two Oracle Database user accounts, which are granted the `DV_OWNER` and `DV_ACCTMGR` roles, respectively.
- Configuring and Enabling Database Vault in the CDB Root
  You can configure and enable Oracle Database Vault with common users who will use the Database Vault-enforced roles in the CDB root.
- Resetting Oracle Database Vault Account Passwords
  Backup accounts can help you reset lost passwords for users who have been granted the `DV_OWNER` and `DV_ACCTMGR` roles.

3.2.4 Configuring and Enabling Database Vault Local Users to Manage Specific PDBs

You must configure and enable Oracle Database Vault in the root first, and then in the PDBs afterward.

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

1. Log in to the PDB as a user who has privileges to create users and to grant the `CREATE SESSION` and `SET CONTAINER` privileges.
2. If necessary, open the database.

   ```
   ALTER DATABASE OPEN;
   ```

3. If you are not using existing local user named accounts for the new Database Vault roles, create new named local user accounts.

In both cases, you must create backup accounts to hold the Database Vault roles in case the named user loses or forgets their password.

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

Oracle strongly recommends that you maintain two accounts for each role. One account, the primary named user account, will be used on a day-to-day basis and the other account will be used as a backup account in case the password of the primary account is lost and must be reset. If you do not have a backup for your account, then you cannot reset passwords. In addition, you must grant the `DV_OWNER` role, `WITH ADMIN OPTION`, to be able to reset the password of the primary account. Store these passwords in a safe location, such as a privileged account management (PAM) system, in case they are needed in the future.

4. Ensure that you have configured and enabled Oracle Database Vault in the CDB root.

   Temporarily connect to the root and then query the `DBA_DV_STATUS` view.

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

   SELECT * FROM DBA_DV_STATUS;
   ```

5. Connect to the PDB as user `SYS` with the `SYSDBA` administrative privilege.
6. While still in the PDB, configure the two backup Database Vault user accounts.

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

   In this example, the `force_local_dvowner` parameter is omitted because it is unnecessary. Database Vault roles are granted locally when configured in a PDB.

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

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

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

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

For example:

```
CONNECT dbv_owner_backup@pdb_name
Enter password: password
```

9. Enable Oracle Database Vault in this PDB.

```
EXEC DBMS_MACADM.ENABLE_DV;
```

10. Connect to the CDB with the `SYSDBA` administrative privilege.

```
CONNECT / AS SYSDBA
```

11. Close and reopen the PDB.

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

12. Verify that the PDB is configured and enabled for Database Vault and Oracle Label Security.

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

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

13. Connect as the backup `DV_OWNER` user and then grant the `DV_OWNER` role, including the `WITH ADMIN OPTION` clause, to the primary `DV_OWNER` user that you created earlier.

    For example:

```
CONNECT dbv_owner_backup@pdb_name
Enter password: password

GRANT DV_OWNER TO sec_admin_owen WITH ADMIN OPTION;
```

14. Connect as the backup `DV_ACCTMGR` user and then grant the `DV_ACCTMGR` role, including the `WITH ADMIN OPTION` clause, to the backup `DV_ACCTMGR` user.

    For example:

```
CONNECT dbv_acctmgr_backup@pdb_name
Enter password: password

GRANT DV_ACCTMGR TO accts_admin_ace WITH ADMIN OPTION;
```

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

Related Topics

- Verifying That Database Vault Is Configured and Enabled
  The `DBA_DV_STATUS`, `CDB_DV_STATUS`, and `DBA_OLS_STATUS` data dictionary views verify if Oracle Database is configured and enabled.
- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.
- Configuring and Enabling Database Vault in the CDB Root
  You can configure and enable Oracle Database Vault with common users who will use the Database Vault-enforced roles in the CDB root.
- Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control
  Oracle Enterprise Manager Cloud Control (Cloud Control) provides pages for managing Oracle Database Vault.

3.2.5 Configuring and Enabling Oracle Database Vault in an Oracle Real Application Clusters Environment

You can configure Oracle Database Vault for an Oracle Real Application Clusters (Oracle RAC) environment, including each Oracle RAC node.

To configure Oracle Database vault for an Oracle RAC environment, you must configure and enable Oracle Database Vault on one node, then restart each of the instance nodes to enable it everywhere. The following procedure assumes that you have a separate Oracle home for each node. If you are familiar with rolling patch installation on Oracle RAC, then you will perform similar steps to enable Oracle Database Vault on all Oracle RAC nodes.

1. Configure and enable Oracle Database Vault in the CDB root.
2. Log into the PDB as user `SYS` with the `SYSDBA` administrative privilege.
3. If necessary, open the database.

   ```
   ALTER DATABASE OPEN;
   ```

4. Run the following `ALTER SYSTEM` statements on either of the Oracle RAC nodes:

   ```
   ALTER SYSTEM SET OS_ROLES=FALSE SCOPE=SPFILE;
   ALTER SYSTEM SET REMOTE_LOGIN_PASSWORDFILE='EXCLUSIVE' SCOPE=SPFILE;
   ALTER SYSTEM SET SQL92_SECURITY=TRUE SCOPE=SPFILE;
   ```

5. Close and then reopen the PDB.

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

Related Topics

- Configuring and Enabling Database Vault in the CDB Root
  You can configure and enable Oracle Database Vault with common users who will use the Database Vault-enforced roles in the CDB root.

3.2.6 Creating a Profile to Protect the DV_OWNER and DV_ACCTMGR Users

A profile provides additional protection for users who have been granted the DV_OWNER and DV_ACCTMGR roles.

Database users who have been granted the DV_OWNER or DV_ACCTMGR roles are considered critical, privileged, accounts. Typically, these accounts should be considered service accounts and exempt from password lockout requirements. Oracle recommends that you create a custom profile that prevents the account from being locked. In addition, you should audit failed login attempts for these Database Vault-related accounts.

1. Log into the database instance as a user who has the CREATE PROFILE system privilege.
   - For common DV_OWNER and DV_ACCTMGR users: Log in to the root of the database instance.
   - For local DV_OWNER and DV_ACCTMGR users: Log in to the PDB in which you created the users.
2. Create a profile similar to the following:
   - For common DV_OWNER and DV_ACCTMGR users: In the root, create the profile similar to the following:

     ```
     CREATE PROFILE c##dv_profile limit
     FAILED_LOGIN_ATTEMPTS UNLIMITED
     PASSWORD_VERIFY_FUNCTION ORA12C_VERIFY_FUNCTION
     PASSWORD_LOCK_TIME UNLIMITED
     CONTAINER=ALL;
     ```

   - For local DV_OWNER and DV_ACCTMGR users: In the PDB, create the profile similar to the following:

     ```
     CREATE PROFILE dv_profile limit
     FAILED_LOGIN_ATTEMPTS UNLIMITED
     PASSWORD_VERIFY_FUNCTION ORA12C_VERIFY_FUNCTION
     PASSWORD_LOCK_TIME UNLIMITED
     CONTAINER=CURRENT;
     ```

3. Update the DV_OWNER and DV_ACCTMGR user accounts to use this profile.
   - For common DV_OWNER and DV_ACCTMGR users:

     ```
     ALTER USER c##sec_admin_owen PROFILE c##dv_profile CONTAINER =
     ALL;
     ALTER USER c##dbv_owner_root_backup PROFILE c##dv_profile
     CONTAINER = ALL;
     ALTER USER c##accts_admin_ace PROFILE c##dv_profile CONTAINER =
     ALL;
     ALTER USER c##dbv_acctmgr_root_backup PROFILE c##dv_profile
     CONTAINER = ALL;
     ```

- For local DV_OWNER and DV_ACCTMGR users:

```
ALTER USER sec_admin_owen PROFILE dv_profile CONTAINER = CURRENT;
ALTER USER dbv_owner_backup PROFILE dv_profile CONTAINER = CURRENT;
ALTER USER accts_admin_ace PROFILE dv_profile CONTAINER = CURRENT;
ALTER USER dbv_acctmgr_backup PROFILE dv_profile CONTAINER = CURRENT;
```

4. Connect as a user who has been granted the AUDIT_ADMIN role.
5. Create and enable a unified audit policy to track failed logins by any user who has been granted the DV_OWNER or DV_ACCTMGR role.
   - For common DV_OWNER and DV_ACCTMGR users: In the root, create a policy similar to the following:

```
CREATE AUDIT POLICY c##dv_logins ACTIONS LOGON;
AUDIT POLICY c##dv_logins BY USERS WITH GRANTED ROLES DV_OWNER,
DV_ACCTMGR
WHENEVER NOT SUCCESSFUL;
```

   - For local DV_OWNER and DV_ACCTMGR users: In the PDB, create a policy similar to the following:

```
CREATE AUDIT POLICY dv_logins ACTIONS LOGON;
AUDIT POLICY dv_logins BY USERS WITH GRANTED ROLES DV_OWNER,
DV_ACCTMGR
WHENEVER NOT SUCCESSFUL;
```

Related Topics

- *Oracle Database SQL Language Reference*
- *Oracle Database Security Guide*

3.2.7 Manually Installing Oracle Database Vault

Under certain conditions, you must manually install Oracle Database Vault.

For example, you may have to manually install Oracle Database Vault if a release 19c Oracle database without Database Vault is upgraded to release 23c, then converted to a PDB to be plugged into a 23c Database Vault-enabled database. In addition,you must manually install Oracle Database Vault (and Oracle Label Security) in a PDB if this PDB does not have these products when the PDB has been plugged into a CDB where Database Vault and Label Security are installed.

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

   Alternatively, log in to the CDB root as a user with DV_OWNER or DV_ADMIN role, and then check that all of the PDBs are open and if Oracle Database Vault is in all of the associated PDBs. You can check if the PDB is open by connecting to it and then querying the OPEN_MODE column from the V$DATABASE view. To find if there is an Oracle Database Vault installation on the CDB, as a user with the SYSDBA administrative privielge, run this query:

```
SELECT * FROM DBA_DV_STATUS;
```

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

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

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

   If properly installed, the result of this should show the major database version and a status of VALID for both Oracle Label Security and Oracle Database Vault:

   ```
   SELECT COMP_NAME, VERSION, STATUS FROM DBA_REGISTRY WHERE COMP_ID
   IN ('DV','OLS');
   ```

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

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

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

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

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

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

After the installation is complete, you can configure and enable Oracle Database Vault in the PDB. After Database Vault is configured and enabled in the CDB root and the database has been restarted, then you can configure and enable Database Vault in the PDB. Database Vault must be configured and enabled in CDB root before it can be configured and enabled in the PDB. After Database Vault is configured and enabled in the CDB root and the database has been restarted, then you can open the PDB and configure and enable Database Vault.

Related Topics

- Configuring and Enabling Oracle Database Vault
  You can configure and enable Oracle Database Vault based on several scenarios.

3.3 Verifying That Database Vault Is Configured and Enabled

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

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

- For Database Vault:

  - If you want to find the Database Vault status for the root only or an individual PDB, then query `DBA_DV_STATUS`. For example:

    ```
    SELECT * FROM DBA_DV_STATUS;
    ```

    Output similar to the following appears:

    ```
    NAME                 STATUS
    -------------------- -----------
    DV_APP_PROTECTION    NOT CONFIGURED
    DV_CONFIGURE_STATUS  TRUE
    DV_ENABLE_STATUS     TRUE
    ```

    `DV_APP_PROTECTION` refers to operations control, which automatically restricts common users from accessing PDB local data in Oracle Database multitenant environments.
  - If you want to find the Database Vault status of all PDBs in the multitenant environment, then as a common user with administrative privileges, query `CDB_DV_STATUS`, which provides the addition of a container ID (`CON_ID`) field.
- For Oracle Label Security, query the `DBA_OLS_STATUS` data dictionary view.

Related Topics

- Using Database Vault Operations Control to Restrict Multitenant Common User Access to Local PDB Data
  You can control PDB access by CDB root common users, such as infrastructure database administrators.

3.4 Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control

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

Only Oracle Enterprise Manager Cloud Control is supported, not Oracle EM Express. The Oracle Database Vault pages can be used to administer and monitor Database Vault-protected databases from a centralized console. This console enables you to automate alerts, view Database Vault reports, and propagate Database Vault policies to other Database Vault-protected databases.

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

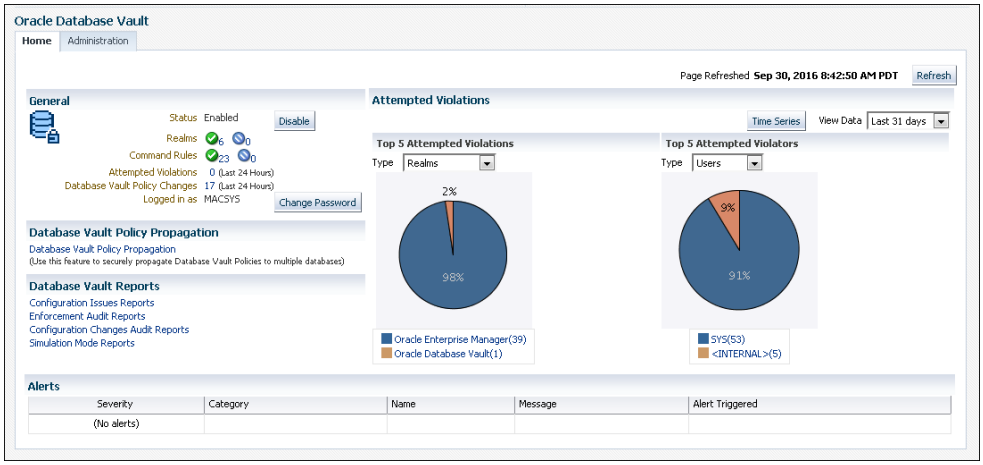
1. Log in to Oracle Enterprise Manager Cloud Control with the credentials that were provided by your Cloud Control administrator.
2. In the Cloud Control home page, from the **Targets** menu, select **Databases**.
3. In the Databases page, select the link for the Oracle Database Vault-protected database to which you want to connect.

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

The Database Login page appears.

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

   The Database Vault home page appears.



Related Topics

- About Oracle Database Vault Roles
  Oracle Database Vault provides a set of roles that are required for managing Oracle Database Vault.
- Using Oracle Database Vault with Oracle Enterprise Manager
  Oracle Database Vault administrators can perform tasks in Oracle Enterprise Manager Cloud Control such as propagating polices to other databases.

3.5 Quick Start Tutorial: Securing a Schema from DBA Access

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

- About This Tutorial
  In this tutorial, you create a realm around the EMPLOYEES table in the HR sample database schema by using the Oracle Database Vault PL/SQL packages.
- Step 1: Log On as SYSTEM to Access the HR Schema
  You must enable the HR schema for this tutorial.
- Step 2: Create a Realm
  Realms can protect one or more schemas, individual schema objects, and database roles.
- Step 3: Create a Unified Audit Policy for Realm Violations
  You can create unified audit policies for Oracle Database Vault realms, rule sets, and factors.
- Step 4: Create the SEBASTIAN User Account
  At this stage, only the HR schema and database users, or roles, with direct object grants can manipulate the database objects the realm will protect. Users relying on system privileges cannot.
- Step 5: Have User SEBASTIAN Test the Realm
  At this stage, user sebastian can test the realm by trying to query the HR.EMPLOYEES table.
- Step 6: Create an Authorization for the Realm
  Next, user SEBASTIAN must be granted authorization to the HR Apps realm, so that they can access the HR.EMPLOYEES table.
- Step 7: Test the Realm
  To test the realm, you must try to access the EMPLOYEES table as a user other than HR.
- Step 8: View Audit Records from Realm Violations
  You should conduct a periodic review of violations on the unified audit policies that you have created.
- Step 9: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

3.5.1 About This Tutorial

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

You will also learn how to create a unified audit policy to record realm violations and review them.

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.

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

You must enable the HR schema for this tutorial.

Before you begin this tutorial, ensure that the HR sample schema is installed.

1. Log in to a PDB as a user who has been granted the `DBA` role, and then access the `HR` schema.
2. Query the `HR.EMPLOYEES` table as follows.

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

Output similar to the following appears:

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

9 rows selected.
```

3. If the `HR` schema is locked and expired, log in to the database instance as the `DV_ACCTMGR` user and unlock and unexpire the account. For example:

```
ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password
```

Replace *password* with a password that meets the password complexity requirements of the user's profile.

A user with the `DBA` role has access to the salary information in the `EMPLOYEES` table of the `HR` schema. Logging in as `SYSTEM` works because `SYSTEM` has the `DBA` role, which has the `SELECT ANY TABLE` system privilege.

4. Do not exit SQL*Plus.

Related Topics

- *Oracle Database Sample Schemas*
- *Oracle Database Security Guide*

3.5.3 Step 2: Create a Realm

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

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

1. Connect to a PDB as a user who has been granted the `DV_OWNER` role.
2. Create the `HR Apps` realm around the `HR.EMPLOYEES` table.
   a. Create the `HR Apps` realm itself.

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'HR Apps',
  description   => 'Realm to protect the HR schema',
  enabled       => DBMS_MACUTL.G_YES,
```

```
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_OFF,
  realm_type    => 0);
END;
/
```

b. Add the `HR.EMPLOYEES` table to this realm.

```
BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'HR Apps',
  object_owner => 'HR',
  object_name  => 'EMPLOYEES',
  object_type  => 'TABLE');
END;
/
```

At this stage, you have created the realm but you have not assigned any authorizations to it. Because this is a traditional realm (`realm_type => 0`), any user with direct grants to `READ` or `SELECT` from `HR.EMPLOYEES` will still be able to view this table, but users who rely on system privileges, such as `READ ANY TABLE` or `SELECT ANY TABLE`, will not. You will take care of that later on in this tutorial.

3.5.4 Step 3: Create a Unified Audit Policy for Realm Violations

You can create unified audit policies for Oracle Database Vault realms, rule sets, and factors.

Unified Audit policies can be designed to create audit records whenever access to realms, rule sets, or factors are not successful, when access is successful, or when access is either successful or not successful. See *Oracle Database Security Guide* for more information about creating unified audit policies for Oracle Database Vault.
After you create a realm, you can create unified audit policies to capture violations of the realm.

1. Connect to a PDB as a user who has been granted the `DBA` role, `AUDIT SYSTEM` system privilege, or `AUDIT_ADMIN` role.

   It may be easier to create a user with this privilege for the tutorial.

   a. As a user with the `DV_ACCTMGR` role, create a temporary user to act as audit administrator.

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

   b. As a user with the `SYSDBA` administrative privilege, grant this new user the `AUDIT_ADMIN` role.

   ```
   GRANT AUDIT_ADMIN TO cmack;
   ```

   c. As a user with the `DV_OWNER` role, grant the new user authorization to use their `AUDIT_ADMIN` role.

   ```
   EXEC DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN('CMACK');
   ```

2. As the `AUDIT_ADMIN` user, `cmack`, create a unified audit policy to audit all actions that are violations of the `HR Apps` realm.

a. Create the unified audit policy.

```
CREATE AUDIT POLICY aud_hrapps_dv
ACTIONS COMPONENT=DV REALM VIOLATION ON "HR Apps";
```

b. Enable the unified audit policy.

```
AUDIT POLICY aud_hrapps_dv;
```

At this stage, you have created the unified audit policy but you have not committed any realm violations yet. You will take care of that later in this tutorial.

3.5.5 Step 4: Create the SEBASTIAN User Account

At this stage, only the `HR` schema and database users, or roles, with direct object grants can manipulate the database objects the realm will protect. Users relying on system privileges cannot.

So, the next step is to authorize database accounts or database roles so that they have access to the schemas or objects protected by the realm. In this tutorial, the only object we have protected is the `HR.EMPLOYEES` table. User `cmack` has been granted the `AUDIT_ADMIN` role, and through the `DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN` authorization that you performed earlier, authorized to use the `AUDIT_ADMIN` role to manage unified audit policies and view unified audit records. You will create the `sebastian` user account.

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

   For example:

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

   Replace *`password`* with a password that meets the password complexity requirements of the user's profile.

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

   ```
   GRANT READ ANY TABLE TO sebastian;
   ```

3. Do not exit SQL*Plus.

Related Topics

- *Oracle Database Security Guide*

3.5.6 Step 5: Have User SEBASTIAN Test the Realm

At this stage, user `sebastian` can test the realm by trying to query the `HR.EMPLOYEES` table.

1. Connect as user `sebastian`.
2. Query the `HR.EMPLOYEES` table.

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

   The following output should appear:

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

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

3.5.7 Step 6: Create an Authorization for the Realm

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

1. Connect to the PDB as a user who has been granted the `DV_OWNER` role.
2. Create an authorization for the `HR Apps` realm.

   This authorization allows `SEBASTIAN` to use the `READ ANY TABLE` system privilege on the `HR.EMPLOYEES` table that is protected by this realm.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name  => 'HR Apps',
  grantee     => 'SEBASTIAN');
END;
/
```

3.5.8 Step 7: Test the Realm

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

(This tutorial does not cover the ability to prevent `HR` from accessing its own objects.) The `SYSTEM` account normally has access to all objects in the `HR` schema because it has the `SELECT ANY TABLE` privilege, but now that you have safeguarded the `EMPLOYEES` table with Oracle Database Vault, this is no longer the case.

1. In SQL*Plus, connect to the PDB as `SYSTEM`.
2. Try querying any of the rows in the `EMPLOYEES` table again.

   For example:

```
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 `EMPLOYEES` table. (In fact, even user `SYS` does not have any access to this table.) However, user `SEBASTIAN` does have access to this information because `sebastian` is an authorized participant in the `HR Apps` realm.
3. Connect as user `sebastian`.

   Perform the following query:

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

   Output similar to the following appears:

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

9 rows selected.
```

3.5.9 Step 8: View Audit Records from Realm Violations

You should conduct a periodic review of violations on the unified audit policies that you have created.

You should create these unified audit policies to track violations to realms, rule sets, and factors. Ideally, you would incorporate this information with a security solution, such as Oracle Data Safe, and send notifications out immediately for all violations.

1. Connect to the PDB as the user `cmack`.

   At minimum, a user must have the `AUDIT_VIEWER` role to view unified audit policy records. User `cmack` has been granted the `AUDIT_ADMIN` role, and through the `DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN` authorization that you performed earlier, authorized to use the `AUDIT_ADMIN` role to manage unified audit policies and view unified audit records.

2. As user `cmack`, query the `UNIFIED_AUDIT_TRAIL` view, which stores the unified audit records.

   ```
   column dbusername format a20
   column action_name format a20
   column object_name format a20
   column object_schema format a20
   column dv_action_object_name format a25

   SELECT DBUSERNAME, ACTION_NAME, OBJECT_SCHEMA, OBJECT_NAME,
   DV_ACTION_OBJECT_NAME, DV_RETURN_CODE
   FROM UNIFIED_AUDIT_TRAIL
   WHERE AUDIT_TYPE = 'Database Vault'
   AND DV_ACTION_NAME = 'Realm Violation Audit'
   ORDER BY EVENT_TIMESTAMP;
   ```

   Output similar to the following appears:

   ```
   DBUSERNAME           ACTION_NAME          OBJECT_SCHEMA
   OBJECT_NAME          DV_ACTION_OBJECT_NAME     DV_RETURN_CODE
   -------------------- -------------------- --------------------
   -------------------- ------------------------- -------------
   SEBASTIAN            SELECT               HR
   EMPLOYEES            HR Apps                             1031
   ```

```
SYSTEM                SELECT                HR
EMPLOYEES             HR Apps                                1031
```

3.5.10 Step 9: Remove the Components for This Tutorial

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

1. As the user `cmack`, disable and drop the `aud_hrapps_dv` unified audit policy.

   ```
   NOAUDIT POLICY aud_hrapps_dv;
   DROP AUDIT POLICY aud_hrapps_dv;
   ```

2. As a user who has the `DV_ACCTMGR` role, drop users `cmack` and `sebastian`.

   ```
   DROP USER cmack;
   DROP USER sebastian;
   ```

3. Delete the `HR Apps` realm.
   a. Connect to the PDB as a user who has been granted the `DV_OWNER` role.
   b. Run the following statement to drop the `HR Apps` realm and its authorizations:

      ```
      EXEC DBMS_MACADM.DELETE_REALM_CASCADE('HR Apps');
      ```

4. If necessary, lock and expire the `HR` account.
   a. Connect as a user who has the `DV_ACCTMGR` role (for example, user `accts_admin_ace`).
   b. Run the following `ALTER USER` statement:

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

4
Configuring Realms

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

- What Are Realms?
  Realms enable you to protect database objects, including specific object types.
- Default Realms
  Oracle Database Vault provides default realms to protect Database Vault and `SYS`-related schemas, system and object privileges, roles, and audit-related objects.
- Creating a Realm
  The first step in enabling realm protection is to create the realm itself, and then add realm-secured objects, roles, and authorizations.
- Modifying a Realm
  You can use the `DBMS_MACADM.UPDATE_REALM` procedure to modify the definition of a realm.
- Deleting a Realm
  You can use the `DBMS_MACADM.DELETE_REALM` procedure to delete a realm.
- About Realm-Secured Objects
  Realm-secured objects define the territory—a set of schema and database objects and roles—that a realm protects.
- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.
- Realm Authorizations in a Multitenant Environment
  The rules and behavior for common realm authorizations are similar to the authorizations for other common objects.
- How Realms Work
  When an appropriately privileged database account issues a SQL statement that affects an object within a realm, a special set of activities occur.
- How Authorizations Work in a Realm
  Realm authorizations prevent users from performing activities if the users do not have the correct privileges.
- Access to Objects That Are Protected by a Realm
  You can protect an object by a realm, but still enable access to objects that are part of this realm-protected object.
- Example of How Realms Work
  Realms can provide protection in which two users who each have the same privileges must have separate access levels for an object.
- How Realms Affect Other Oracle Database Vault Components
  Realms have no effect on factors, identities, or rule sets, but they do affect command rules.

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

4.1 What Are Realms?

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

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

4.1.1 About Realms

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

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

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

For example, you can create a realm to protect the database schemas that are used by an accounting department's application. The realm will prohibit any user from using their system privileges (for example, `SELECT ANY TABLE`) from accessing the realm-

protected schema objects. When an entire schema is realm protected, all existing and new objects are protected. This includes tables, indexes, procedures, views, packages, and more. Direct object grants will be allowed in regular realms but not in mandatory realms. Mandatory realms require the granted user to be an authorized participant in the mandatory realm.

You can run reports on realms that you create in Oracle Database Vault. Realms can be enabled, disabled, or placed in simulation mode where violations of the realm will be logged but the action will not be blocked. This enables you to quickly test applications using Database Vault realms.

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

4.1.2 Mandatory Realms to Restrict User Access to Objects within a Realm

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

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

Mandatory realms have the following additional characteristics:

- If a role is protected by a mandatory realm, then no privileges can be granted to or revoked from the protected role except by the realm owner.
- You can update regular realms that you created in earlier releases to be mandatory realms. This way, you can block owner access and object-privileged users from accessing the realm-protected objects.
- `SYS`-owned objects are already protected by data dictionary protection and are not protected separately by Oracle Database Vault.

Mandatory realms have the following benefits:

- **Mandatory realms can block object owners and object privileged users.** In previous releases, blocking these users could only be done by defining complicated command rules.
- **Mandatory realms provide more flexible configurations for access control.** For example, suppose you want to enable a user to access an object with certain conditions, such as in a specific time range during the day. You cannot grant object privileges to that user because realms do not block object privileges. You only can grant system privileges to the user and then authorize this user to the realm with a rule, or make a command rule on the command directly. These solutions are either very expensive in terms of computational cost or undesirable because they entail the excessive granting of privileges such as system privileges to the user. With a mandatory realm, you only need to grant object privileges to the user, with a rule for specific conditions, and then authorize this user to be a realm owner or participant. Thus, with mandatory realms, Oracle Database Vault policies have more flexibility without granting users excessive privileges.
- **Mandatory realms add a layer of protection during patch upgrades.** During a patch upgrade, a database administrator may need to have direct access to a realm-protected object in order to perform a patch on the object. If there are tables that contain sensitive data, such as social security numbers, you can protect these tables from the

administrator's access with mandatory realms during the patch upgrade. When patching is complete, and the database administrator no long needs access to the objects, you can disable mandatory realm protection and then re-enable the normal application realm protection so that the application protection can return to its normal state.

- **You can use mandatory realms to secure tables during runtime.** During runtime, application data can be stored in many tables. It is better to have a single user such as a runtime schema to access these tables so that you can maintain the integrity and correctness of the data. If the application data is scattered in many different schemas, then schema owners and users with object privileges can change the data if they log in to the database directly. To insure that users cannot update these tables without going through the runtime schema's procedures, you can use mandatory realms to protect the tables so that only the authorized user's procedures can access them. Because a regular realm does not block object owners and object-privileged users, you can use mandatory realms to block them. This way, only authorized users can access these tables during runtime.

If there are multiple mandatory realms on the same object, then you must authorize the user or role on all the mandatory realms before they can access the protected object.

Related Topics

- CREATE_REALM Procedure
  The `CREATE_REALM` procedure creates both common and local realms.
- UPDATE_REALM Procedure
  The `UPDATE_REALM` procedure updates a realm.

4.1.3 Realms in a Multitenant Environment

You can create a realm to protect common objects in the application root.

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

You cannot create a common realm in the CDB root.

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

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

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

Related Topics

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

4.1.4 Object Types That Realms Can Protect

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

These object types are as follows:

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

4.2 Default Realms

Oracle Database Vault provides default realms to protect Database Vault and `SYS`-related schemas, system and object privileges, roles, and audit-related objects.

You can add users to realms so that the user can perform tasks that are protected by the default realms.

- Oracle Database Vault Realm
  The `Oracle Database Vault` realm protects configuration and role information in the Oracle Database Vault `DVSYS`, `DVF`, and `LBACSYS` schemas.
- Database Vault Account Management Realm
  The `Database Vault Account Management` realm defines the realm for the administrators who manage and create database accounts and database profiles.
- Oracle Enterprise Manager Realm
  Oracle Database Vault provides a realm specifically for Oracle Enterprise Manager monitoring accounts.
- Oracle Default Schema Protection Realm
  The `Oracle Default Schema Protection Realm` protects roles and schemas that are used with Oracle features such as Oracle Text.
- Oracle System Privilege and Role Management Realm
  The `Oracle System Privilege and Role Management Realm` protects all Oracle-supplied roles in an Oracle database.
- Oracle Default Component Protection Realm
  The `Oracle Default Component Protection Realm` protects the `SYSTEM` and `OUTLN` schemas.

- Oracle Label Security Realm
  The `Oracle Label Security` realm protects Oracle Label Security schemas and roles.
- Oracle GoldenGate Protection Realm
  The `Oracle GoldenGate Protection Realm` protects GoldenGate-related objects in the Oracle database.
- Oracle Audit Realm
  The `Oracle Audit` realm protects the `AUDSYS` schema and audit-related objects in the `SYS` schema.

4.2.1 Oracle Database Vault Realm

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

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

To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'Oracle Database Vault Realm'
ORDER BY OWNER, OBJECT_NAME;
```

To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Oracle Database Vault Realm'
ORDER BY GRANTEE;
```

Related Topics

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

4.2.2 Database Vault Account Management Realm

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

The owner of this realm can grant or revoke the `CREATE SESSION` privilege to or from a user.

To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
```

```
WHERE REALM_NAME = 'Database Vault Account Management'
ORDER BY OWNER, OBJECT_NAME;
```

To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Database Vault Account Management'
ORDER BY GRANTEE;
```

Related Topics

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

4.2.3 Oracle Enterprise Manager Realm

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

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

To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'Oracle Enterprise Manager'
ORDER BY OWNER, OBJECT_NAME;
```

To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Oracle Enterprise Manager'
ORDER BY GRANTEE;
```

Related Topics

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

4.2.4 Oracle Default Schema Protection Realm

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

The advantage of this grouping is that Oracle Spatial schemas (`MDSYS`, `MDDATA`) are used extensively with Oracle Text (`CTXSYS`).

`Oracle Default Schema Protection Realm` protects several roles and schemas.

- To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'Oracle Default Schema Protection Realm'
ORDER BY OWNER, OBJECT_NAME;
```

- To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Oracle Default Schema Protection Realm'
ORDER BY GRANTEE;
```

- Roles that are protected by default: `CTXAPP`, `EJBCLIENT`
- Schemas that are protected by default: `CTXSYS`, `EXFSYS`, `MDDATA`, `MDSYS`
- Schemas that are recommended for protection: `APEX_030200`, `OWBSYS`, `WMSYS`

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

4.2.5 Oracle System Privilege and Role Management Realm

The `Oracle System Privilege and Role Management Realm` protects all Oracle-supplied roles in an Oracle database.

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

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

Examples of roles that the `Oracle System Privilege and Role Management Realm` protects are `DBA`, `IMP_FULL_DATABASE`, `SELECT_CATALOG_ROLE`, and `SCHEDULER_ADMIN`.

To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'Oracle System Privilege and Role Management Realm'
ORDER BY OWNER, OBJECT_NAME;
```

To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Oracle System Privilege and Role Management Realm'
ORDER BY GRANTEE;
```

4.2.6 Oracle Default Component Protection Realm

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

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

To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'Oracle Default Component Protection Realm'
ORDER BY OWNER, OBJECT_NAME;
```

To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Oracle Default Component Protection Realm'
ORDER BY GRANTEE;
```

4.2.7 Oracle Label Security Realm

The `Oracle Label Security` realm protects Oracle Label Security schemas and roles.

By default, the `LBACSYS` and `DVSYS` users are the owners for the realm, and the `LBAC_DBA` role is authorized to the realm as a participant. `LBACSYS` is the schema for this realm.

To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'Oracle Label Security'
ORDER BY OWNER, OBJECT_NAME;
```

To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Oracle Label Security'
ORDER BY GRANTEE;
```

4.2.8 Oracle GoldenGate Protection Realm

The `Oracle GoldenGate Protection Realm` protects GoldenGate-related objects in the Oracle database.

By default, the `GGSHAREDCAP` user is the owner of the realm.

To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'Oracle GoldenGate Protection Realm'
ORDER BY OWNER, OBJECT_NAME;
```

To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Oracle GoldenGate Protection Realm'
ORDER BY GRANTEE;
```

4.2.9 Oracle Audit Realm

The `Oracle Audit` realm protects the `AUDSYS` schema and audit-related objects in the `SYS` schema.

The audit-related objects include `SYS`-owned tables (such as `SYS.AUD$`), views (such as `AUDSYS.UNIFIED_AUDIT_TRAIL`, and packages (such as `DBMS_AUDIT_MGMT`). Oracle Database Vault does not allow attempts to alter, drop, or replace any of these objects, and generates an audit record for each attempt. By default, only the `AUDDYS` schema is authorized to the `Oracle Audit` realm, and the `SYS` user who is granted with the `DV_PATCH_ADMIN` role during patching can modify the protected objects.

To find the objects that this realm protects, perform the following query:

```
SELECT OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'Oracle Audit'
ORDER BY OWNER, OBJECT_NAME;
```

To find the realm-authorized users, their role as participant or owner, and if an Oracle Database Vault rule set is applied to the authorized user, perform the following query:

```
SELECT GRANTEE, AUTH_OPTIONS, AUTH_RULE_SET_NAME
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'Oracle Audit'
ORDER BY GRANTEE;
```

Related Topics

- Using Oracle Database Auditing with Oracle Database Vault
  You grant and revoke authorizations to manage and view audit records in an Oracle Database Vault environment.

4.3 Creating a Realm

The first step in enabling realm protection is to create the realm itself, and then add realm-secured objects, roles, and authorizations.

1. Connect to the PDB or the application root as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. Execute the DBMS_MACADM.CREATE_REALM procedure to create the realm.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'HR Realm',
  description   => 'Realm to protect the HR schema',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_OFF,
  realm_type    => 1,
  realm_scope   => DBMS_MACUTL.G_SCOPE_LOCAL,
  pl_sql_stack  => TRUE);
END;
/
```

   In this specification:

   - realm_name can be up to 128 characters in mixed-case. Oracle suggests that you use the name of the protected application as the realm name (for example, hr_app for an human resources application). This parameter is mandatory. The DBA_DV_REALM data dictionary view lists existing realms.
   - description can be 1024 characters in mixed-case. You may want to include a description for the business objective of the given application protection and document all other security policies that compliment the realm's protection. Also document who is authorized to the realm, for what purpose, and any possible emergency authorizations.
   - enabled controls realm checking. Valid settings are DBMS_MACUTL.G_YES 'y' to enable realm checking (default), DBMS_MACUTL.G_NO or 'n' to disable all realm checking, including the capture of violations in the simulation log, or DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to run but capture violations in the simulation log.
   - audit_options applies only to traditional auditing, not unified auditing environments. Starting with Oracle Database release 23c, traditional auditing is desupported. Only the DBMS_MACUTL.G_REALM_AUDIT_OFF setting is available for the audit_options parameter. To audit a realm, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a realm.
   - realm_type defines whether the realm is mandatory (1) or not mandatory (0). When set to mandatory, only realm owners or realm participants will have access to objects in a realm. Object owners and object-privileged users who are not realm owners or participants will have no access.
   - realm_scope defines whether the realm is created in a PDB (DBMS_MACUTL.G_SCOPE_LOCAL) or in an application root (DBMS_MACUTL.G_SCOPE_COMMON). If you create the common realm in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example:

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

   - pl_sql_stack is used for simulation mode, and when enabled (TRUE), it specifies whether to record the PL/SQL stack for failed operations. To disable, enter FALSE. The default is FALSE.

At this stage, the realm is created, but it protects no objects nor does it have any authorizations.

3. Run the `DBMS_MACADM.ADD_OBJECT_TO_REALM` procedure to add objects (such as tables or roles) to the realm so that they can be protected.

   For example:

   ```
   BEGIN
    DBMS_MACADM.ADD_OBJECT_TO_REALM(
     realm_name   => 'HR Realm',
     object_owner => 'HR',
     object_name  => 'EMPLOYEES',
     object_type  => 'TABLE');
   END;
   /
   ```

   In this specification:

   - `realm_name` can be up to 128 characters in mixed-case.
   - `object_owner` is the owner of the object that is being added to a realm. You can enter the `%` character if the object you want to secure with the realm is a role.
   - `object_name` is the name of the object that the realm will protect. Alternatively, enter `%` to specify all objects (except roles) for the object owner that you have specified. If you enter `%`, then it can encompass all objects in the schema if `%` is also used for the `object_type` parameter. But if `object_type` is set to `TABLE`, then using `%` for the `object_name` refers to all tables in the schema. Note that the `%` wildcard character applies to objects that do not yet exist and currently existing objects.
   - `object_type` is the type of object, such as `TABLE`, `INDEX`, or `ROLE`. To create a realm for all types, enter `%` or `DBMS_MACUTL.G_ALL_OBJECT`. You can add as many objects of any type as you want to the realm.

4. Run the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure to authorize users for the realm.

   For example:

   ```
   BEGIN
    DBMS_MACADM.ADD_AUTH_TO_REALM(
     realm_name    => 'HR Realm',
     grantee       => 'HR',
     rule_set_name => 'Enabled',
     auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER,
     auth_scope    => DBMS_MACUTL.G_SCOPE_LOCAL);
   END;
   /
   ```

   In this specification:

   - `realm_name` can be up to 128 characters in mixed-case.
   - `grantee` is the user or role name to authorize as an owner or a participant. To find the existing users and roles in the current database instance, query the `DBA_USERS` and `DBA_ROLES` views. To find the authorization of a particular user or role, query the `DVA_DV_REALM_AUTH` view. To find existing secure application roles used in privilege management, query the `DBA_DV_ROLE` view.

- `rule_set_name` is an optional rule set to check during runtime. The `DBA_DV_RULE_SET` data dictionary view lists available rule sets. You can only specify one rule set, but this rule set can have multiple rules.
- `auth_options` determines how to authorize a realm. Valid settings are as follows:
  - `DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT` allows system or direct privileges to access, manipulate, and create objects protected by the realm, provided these rights have been granted using the standard Oracle Database privilege grant process. (Default)
  - `DBMS_MACUTL.G_REALM_AUTH_OWNER` has the same authorization as the realm participant, plus the authorization to grant or revoke realm-secured roles and privileges on realm-protected objects.

  A realm can have multiple participants or owners.
- `auth_scope` defines whether the realm is authorized locally in the current PDB (`DBMS_MACUTL.G_SCOPE_LOCAL`) or in an application root (`DBMS_MACUTL.G_SCOPE_COMMON`).

Related Topics

- Oracle Database Vault Realm APIs
  The `DBMS_MACADM` PL/SQL package enables you to configure Oracle Database Vault realms.
- About Realm-Secured Objects
  Realm-secured objects define the territory—a set of schema and database objects and roles—that a realm protects.
- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

4.4 Modifying a Realm

You can use the `DBMS_MACADM.UPDATE_REALM` procedure to modify the definition of a realm.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Find the realm name and check its definition.

   For example:

   ```
   SELECT NAME, DESCRIPTION, ENABLED, AUDIT_OPTIONS, REALM_TYPE
   FROM DBA_DV_REALM ORDER BY NAME;
   ```

   If you want to change the `ENABLED` setting, then note the following: If the realm is managed by a policy, and if the policy status is set to partial, then you can modify the enablement status of the realm. If the policy is set to enabled, disabled, or simulation mode, then you cannot modify the enablement status of the realm.

3. Run the `DBMS_MACADM.UPDATE_REALM` statement.

   For example:

```
BEGIN
 DBMS_MACADM.UPDATE_REALM(
  realm_name    => 'HR Realm',
  description   => 'Realm to protect the HR schema',
  enabled       => DBMS_MACUTL.G_YES,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_OFF,
  realm_type    => 1);
END;
/
```

   After you update a realm, if it has traditional auditing set (`audit_options`), then this auditing disabled because traditional auditing is desupported starting in release 23c. Only the `DBMS_MACUTL.G_REALM_AUDIT_OFF` setting is available for the `audit_options` parameter. To capture new audit records, create and enable a unified audit policy. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a realm.

Related Topics

- Oracle Database Vault Realm APIs
  The `DBMS_MACADM` PL/SQL package enables you to configure Oracle Database Vault realms.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

4.5 Deleting a Realm

You can use the `DBMS_MACADM.DELETE_REALM` procedure to delete a realm.

When you delete a realm, all the associations that were created for the realm are dropped, too.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Find the names of the realms that you want to remove.

```
SELECT NAME FROM DBA_DV_REALM
ORDER BY NAME;
```

3. Optionally, check the realm's definitions before you decide to delete the realm.
   - To check for any object references to realm, query the `DBA_DV_REALM_OBJECT` data dictionary view. For example:

```
SELECT OBJECT_OWNER, OBJECT_NAME, OBJECT_TYPE
FROM DBA_DV_REALM_OBJECT
WHERE REALM_NAME = 'HR Realm';

OBJECT_OWNER  OBJECT_NAME  OBJECT_TYPE
------------  -----------  ------------
HR            EMPLOYEES    TABLE
```

     If you want to only remove objects from the realm, then you can run the `DBMS_MACADM.DELETE_OBJECT_FROM_REALM` procedure.

- To find the authorizations for the realm, query the `DBA_DV_REALM_AUTH` data dictionary view. For example:

```
SELECT GRANTEE, AUTH_OPTIONS
FROM DBA_DV_REALM_AUTH
WHERE REALM_NAME = 'HR Realm';

GRANTEE  AUTH_OPTIONS
-------  ------------------------
HR       DBMS_MACUTL.G_SCOPE_LOCAL
```

  You can remove authorizations by running the `DBMS_MACADM.DELETE_AUTH_FROM_REALM` procedure.

- To find policies that are associated with the realm, query the `DBA_DV_POLICY_OBJECT` data dictionary view. For example:

```
SELECT POLICY_NAME, COMMAND_OBJ_NAME
FROM DBA_DV_POLICY_OBJECT
WHERE COMMAND_OBJ_NAME = 'HR Realm';
```

  You can run the `DBMS_MACADM.DELETE_REALM_FROM_POLICY` to remove the realm from the policy.

4. Run the `DBMS_MACADM.DELETE_REALM` procedure to delete the realm.

   For example:

```
EXEC DBMS_MACADM.DELETE_REALM('HR Realm');
```

Related Topics

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

4.6 About Realm-Secured Objects

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

You can create the following types of protections:

- Objects from multiple database accounts or schemas can be under the same realm.
- One object can belong to multiple realms.

  If an object belongs to multiple realms, then Oracle Database Vault checks the realms for the proper authorization. For `SELECT`, DDL, and DML statements, as long as a user is a participant in one of the realms, and if the command rules permit it, then the commands that the user enters are allowed. For `GRANT` and `REVOKE` operations of a database role in multiple realms, the person performing the `GRANT` or `REVOKE` operation must be the realm owner. Schema owners can perform DML operations on objects that are protected by multiple regular realms.

  If one of the realms is a mandatory realm, then the user who wants to access the object must be a realm owner or participant in the mandatory realm. During the authorization checking process, the non-mandatory realms are ignored. If there are multiple mandatory realms that protect the object, then the user who wants to access the object must be authorized in all of the mandatory realms.

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

4.7 About Realm Authorization

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

You can grant a realm authorization to an account or role to allow the use of its system privileges in the following situations:

- When the user must create or access realm-secured objects
- When a user must grant or revoke realm-secured roles

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

Note the following:

- Realm owners cannot add other users to their realms as owners or participants. Only users who have the DV_OWNER or DV_ADMIN role are allowed to add users as owners or participants to a realm.
- Users who have been granted the DV_OWNER or DV_ADMIN role can add themselves to a realm authorization.
- A realm owner, but not a realm participant, can grant or revoke realm secured roles or grant or revoke object privileges on realm secured objects to anyone.
- A user can be granted either as a realm owner or a realm participant, but not both. However, you can update the authorization types of existing realm authorizations.

Related Topics

- Realm Authorization Configuration Issues Report
  The Realm Authorization Configuration Issues Report displays Oracle Database Vault realm configuration issues.

4.8 Realm Authorizations in a Multitenant Environment

The rules and behavior for common realm authorizations are similar to the authorizations for other common objects.

Local Authorization for a Common Realm

The local authorization for a common realm refers to the authorization a user has for the PDB that this user is accessing.

The rules for the local authorization for a common realm are as follows:

- A user who has been commonly granted the DV_OWNER or DV_ADMIN role can grant local authorization to common users, common roles, local users, and local roles. The common DV_OWNER or DV_ADMIN user can also remove local authorization from a common realm in a PDB.
- A local Database Vault administrator can authorize locally (that is, grant local authorizations to both local and common users) within the PDB. A common Database Vault administrator can also grant authorizations in each PDB. A

common realm authorization can only be granted by a common Database Vault administrator in the application root.

- The common Database Vault administrator can both add or remove local authorization to and from a common realm from within the PDB.
- If a common user has only local authorization for a common realm, then this user cannot access the common realm in any other PDB than this local authorization.
- A common user or a common role can have both the local authorization and the common authorization to a common realm at the same time. Removing a common user's local authorization from a common realm does not affect the common user's common authorization. Removing a common user's common authorization from a common realm does not affect the common user's local authorization.

Common Authorization for a Common Realm

The common authorization for a common realm refers to the authorization a common user or a common role has in the application root while the authorization takes effect in every container that is Database Vault enabled.

The rules for the local authorization for a common realm are as follows:

- A user who has been commonly granted the `DV_OWNER` or `DV_ADMIN` role can grant common realm authorization to common users or roles in the application root. This common Database Vault administrator can perform the removal of common authorizations while in the application root.
- This common authorization applies to the containers that have been Database Vault enabled in the CDB.
- If a common user is authorized to a common realm in the application root, then this user has access to the objects protected by the common realm in the application root and any application PDBs.
- Any rule sets that are associated with a common realm must be common rule sets. The rules that are added to a common rule set that is associated with common authorization cannot involve any local objects.

How the Authorization of a Realm Works in Both the Application Root and in an Individual PDB

During the Database Vault enforcement in a container, a common realm performs the same enforcement behaviors as the same realm when it is used locally in a PDB.

4.9 How Realms Work

When an appropriately privileged database account issues a SQL statement that affects an object within a realm, a special set of activities occur.

These privileges include DDL, DML, `EXECUTE`, `GRANT`, `REVOKE`, or `SELECT` privileges.

1. Are the user's object privileges correct?
   Oracle Database Vault first checks the user's privileges before allowing the user to continue. If the user does not have the correct privileges, then grant these to the user. If the user's privileges are correct, then go to Step 2. Realm authorization does not implicitly grant additional privileges to the user.
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 8. 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 realm a mandatory realm or regular realm?

   If yes, then go to Step 5. If it is regular realm, then go to Step 4.

4. Is the database account using a system privilege to run the SQL statement?

   If yes, then go to Step 5. If no, then go to Step 7. If the session has object privileges on the object in question for `SELECT`, `EXECUTE`, and DML statements only, then the realm protection is not enforced. Realms protect against the use of any system privilege on objects or roles protected by the realm. Even users with object privileges for objects that are protected by regular realms are prevented from performing DDL operations.

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

   If yes, then go to Step 6. Otherwise, a realm violation occurs and the statement is not allowed to succeed. If the command is a `GRANT` or `REVOKE` of a role that is protected by the realm, or the `GRANT` or `REVOKE` of an object privilege on an object protected by the realm, then the session must be authorized as the realm owner directly or indirectly through roles.

6. Is the realm authorization for the database account conditionally based on a rule set?

   If yes, then go to Step 7. If no, then go to Step 8.

7. Does the rule set evaluate to `TRUE`?

   If yes, then go to Step 8. If no, then there is a realm violation, so the SQL statement is not allowed to succeed.

8. Does a command rule prevent the command from executing?

   If yes, then there is a command rule violation and the SQL statement fails. If no, then there is no realm or command rule violation, so the command succeeds.

   For example, the `HR` account may have the `DROP ANY TABLE` privilege and may be the owner of the `HR` realm, but a command rule can prevent `HR` from dropping any tables in the `HR` schema unless it is during its monthly maintenance window. Command rules apply to the use of the `ANY` system privileges and object privileges and are evaluated after the realm checks.

In addition, because a session is authorized in a realm, it does not mean the account has full control on objects protected by the realm. Realm authorization does *not* implicitly grant extra privileges to the account. The account still must have system privileges or object privileges to access the objects. For example, an account or role may have the `SELECT ANY` table privilege and be a participant in the `HR` realm. This means the account or the account granted the role could query the `HR.EMPLOYEES` table. Being a participant in the realm does not mean the account or role can `DROP` the `HR.EMPLOYEES` table. Oracle Database Vault does not replace the discretionary access control model in the existing Oracle database. It functions as a layer on top of this model for both realms and command rules.

Note the following:

- Protecting a table in a realm does not protect the view by default. Any view that must be protected should be added to the realm regardless of whether the view was created before or after the table was added to the realm.

- For invoker's right procedures that access realm protected objects, the invoker of the procedure must be authorized to the realm.
- Be aware that realm protection does not protect a table if access to the table has been granted to `PUBLIC`. For example, if `SELECT ON table_name` is granted to `PUBLIC`, then every user has access to `table_name` (unless the table is protected by a mandatory realm), even if this table is protected by a realm. As a best practice, revoke unnecessary privileges from `PUBLIC`.

4.10 How Authorizations Work in a Realm

Realm authorizations prevent users from performing activities if the users do not have the correct privileges.

- About Authorizations in a Realm
  Realms protect data from access through system privileges.
- Examples of Realm Authorizations
  You can create realms that protect objects from users who have system privileges and other powerful privileges, for example.

4.10.1 About Authorizations in a Realm

Realms protect data from access through system privileges.

Realms do not give additional privileges to the data owner or participants.

The realm authorization provides a run-time mechanism to check logically if a user's command should be allowed or denied to access objects specified in the command and to proceed with its execution.

System privileges are sweeping database privileges such as `CREATE ANY TABLE` and `DELETE ANY TABLE`. These privileges typically apply across schemas and bypass the need for object privileges. Data dictionary views such as `DBA_SYS_PRIVS`, `USER_SYS_PRIVS`, and `ROLE_SYS_PRIVS` list the system privileges for database accounts and roles. Database authorizations work normally for objects not protected by a realm. However, a user must be authorized as a realm owner or participant to successfully use their system privileges on objects secured by the realm. A realm violation prevents the use of system privileges and can be audited.

Mandatory realms block both object privileged-based access and system privilege-based access. This means that even the object owner cannot have access if they are not authorized to access the realm. Users can access secured objects in the mandatory realm only if the user or role is authorized to do so.

4.10.2 Examples of Realm Authorizations

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

- Example: Unauthorized User Trying to Create a Table
  The `ORA-47401` error appears when unauthorized users try to create tables.
- Example: Unauthorized User Trying to Use the DELETE ANY TABLE Privilege
  An `ORA-01031: insufficient privileges` error appears for unauthorized user access.

- Example: Authorized User Performing DELETE Operation
  Authorized users are allowed to perform the activities for which they are authorized.

4.10.2.1 Example: Unauthorized User Trying to Create a Table

The `ORA-47401` error appears when unauthorized users try to create tables.

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

```
ORA-47401: Realm violation for CREATE TABLE on HR.DEMO2
```

As you can see, the attempt by the unauthorized user fails. Unauthorized use of system privileges such as `SELECT ANY TABLE`, `CREATE ANY TABLE`, `DELETE ANY TABLE`, `UPDATE ANY TABLE`, `INSERT ANY TABLE`, `CREATE ANY INDEX`, and others results in failure.

4.10.2.2 Example: Unauthorized User Trying to Use the DELETE ANY TABLE Privilege

An `ORA-01031: insufficient privileges` error appears for unauthorized user access.

Example 4-2 shows what happens when an unauthorized database account tries to use their `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.

4.10.2.3 Example: Authorized User Performing DELETE Operation

Authorized users are allowed to perform the activities for which they are authorized.

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.
```

4.11 Access to Objects That Are Protected by a Realm

You can protect an object by a realm, but still enable access to objects that are part of this realm-protected object.

For example, suppose you create a realm around a specific table. However, you want users to be able to create an index on this table. You can accomplish this as follows, depending on the following scenarios.

- **The user does not have the CREATE ANY INDEX privilege.** As the realm owner of the table, grant the `CREATE INDEX ON` *`table`* privilege to the user who must create the index.
- **The user has the CREATE ANY INDEX privilege.** In this case, create another realm and make all index types as the secured objects and grant that user participant authorization to the realm. (Remember that having the `CREATE ANY INDEX` privilege alone is not sufficient for a non-realm participant to create an index in a realm-protected table.)
- **You want all of your database administrators to be able to create an index and they have the CREATE ANY INDEX privilege.** In your data protection realm, specify all object types to be protected *except* the index types. This permits all of your administrators to create indexes for the protected table.

4.12 Example of How Realms Work

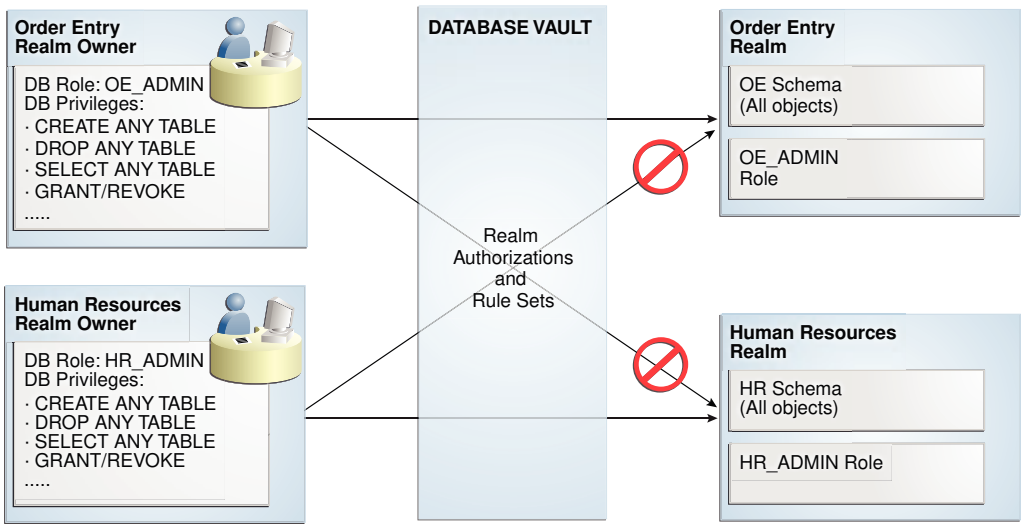
Realms can provide protection in which two users who each have the same privileges must have separate access levels for an object.

Figure 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 System Privilege and Role Management` realm. Any unauthorized attempt to use a system privilege to access realm-protected objects raises a realm violation, which can be audited. The powers of each realm owner are limited within the realm itself. For example, `OE_ADMIN` has no access to the Human Resources realm, and `HR_ADMIN` has no access to the Order Entry realm.

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



Related Topics

- Quick Start Tutorial: Securing a Schema from DBA Access
  This tutorial shows how to create a realm around the `HR` schema.

4.13 How Realms Affect Other Oracle Database Vault Components

Realms have no effect on factors, identities, or rule sets, but they do affect command rules.

With command rules, Oracle Database Vault evaluates the realm authorization first when processing SQL statements.

How Realms Work explains the steps that Oracle Database Vault takes to process SQL statements that affect objects in a realm. How Command Rules Work describes how command rules are processed.

4.14 Guidelines for Designing Realms

Oracle provides a set of guidelines for designing realms.

- Create realms based on the schemas and roles that form a database application.

  Define database roles with the minimum and specific roles and system privileges required to maintain the application objects and grant the role to named accounts. You then can add the role as an authorized member of the realm. For object-level privileges on objects protected by the realm and required by an application, create a role and grant these minimum and specific object-level privileges to the role, and then grant named accounts this role. In most cases, these types of roles do not need to be authorized in the realm unless `ANY`-style system privileges are already in use. A model using the principle of least privilege is ideal for any database application.

- A database object can belong to multiple realms and an account or role can be authorized in multiple realms.

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

- If you want to add a role to a realm as a grantee, create a realm to protect the role. Doing so prevents users who have been granted the `GRANT ANY ROLE` system privilege, such as the `SYSTEM` user account, from granting the role to themselves.

- If you want to add the `SYS` user account to a realm authorization, you must add user `SYS` explicitly and not through a role (such as the `DBA` role).

- Be mindful of the privileges currently allowed to a role that you plan to add as a realm authorization.

  Realm authorization of a role can be accidentally granted and not readily apparent if an account such as `SYS` or `SYSTEM` creates a role for the first time and the Oracle Database Vault administrator adds this role as a realm authorization. This is because the account that creates a role is implicitly granted the role when it is created.

- Sometimes you must temporarily relax realm protections for an administrative task. Rather than disabling the realm, have the Security Manager (`DV_ADMIN` or `DV_OWNER`) log in, add the named account to the authorized accounts for the realm, and set the authorization rule set to Enabled. Then in the enabled rule set, turn on all auditing for the rule set. You can remove the realm authorization when the administrative task is complete.

- If you want to grant `ANY` privileges to new users, Oracle recommends that you add a database administrative user to the Oracle System Privilege and Role Management realm so that this user can grant other users `ANY` privileges, if they need them. For example, using a named account to perform the `GRANT` of the `ANY` operations enables you to audit these operations, which creates an audit trail for accountability.

- If you drop a table, index, or role that has been protected by a realm and then recreate it using the same name, the realm protection is not restored. You must re-create the realm protection for the new table, index, or role. However, you can automatically enforce protection for all future tables, indexes, and roles within a specified schema. For example, to enforce protection for all future tables:

  ```
  BEGIN
   DBMS_MACADM.ADD_OBJECT_TO_REALM('realm_name', 'schema_name', '%', 'TABLE');
  END;
  /
  ```

- You can test the development phase of a realm by using simulation mode, which enables the realm without enforcing the restrictions. Simulation mode writes detailed information about violations, allowing you to see the activities that have been enforced. A user who has the `DV_OWNER` or `DV_ADMIN` role can view the simulation log by querying the `DBA_DV_SIMULATION_LOG` data dictionary view.

Related Topics

- Using Simulation Mode for Logging Realm and Command Rule Activities
  Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

4.15 How Realms Affect Performance

Realms can affect database performance in a variety situations, such as with DDL and DML operations.

- **DDL and DML operations on realm-protected objects do not have a measurable effect on Oracle Database.** Oracle recommends that you create the realm around the entire schema, and then authorize specific users to perform only specific operations related to their assigned tasks. For finer-grained control, you can define realms around individual tables and authorize users to perform certain operations on them, and also have a realm around the entire schema to protect the entire application. Note that this type of configuration (that is, multiple realms protecting the same objects) does not result in significant performance degradation, and it does enable you to grant realm authorization to some of the objects in a schema.
- **Auditing affects performance.** To achieve the best performance, Oracle recommends that you use fine-grained auditing rather than auditing all operations.
- **Periodically check the system performance.** You can do so by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Cloud Control, which is installed by default with Oracle Database), Automatic Workload Repository (AWR), and `TKPROF`.

Related Topics

- *Oracle Database Performance Tuning Guide*
- *Oracle Database SQL Tuning Guide*

4.16 Realm Related Reports and Data Dictionary Views

Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing realms.

Table 4-1 lists the Oracle Database Vault reports.

Table 4-1 Oracle Enterprise Manager Reports Related to Realms

Report	Purpose
Realm Audit Report	Audits records generated by the realm protection and realm authorization operations
Realm Authorization Configuration Issues Report	Lists authorization configuration information, such as incomplete or disabled rule sets, or nonexistent grantees or owners that may affect the realm
Rule Set Configuration Issues Report	Lists rule sets that do not have rules defined or enabled, which may affect the realms that use them
All object privilege reports	List object privileges that the realm affects
Privilege management summary reports	Provide information about grantees and owners for a realm
Sensitive objects reports	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	Lists the realms created in the current database instance.
DBA_DV_REALM_AUTH	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	Lists the database schemas, or subsets of schemas with specific database objects contained therein, that are secured by the realms

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

5
Configuring Rule Sets

Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.

- What Are Rule Sets?
  A rule set is a collection of one or more rules.
- Rule Sets and Rules in a Multitenant Environment
  You can create a rule set and its associated rules in a PDB or an application root.
- Default Rules and Rule Sets from Releases Earlier Than Release 12.2
  Many default rules and rule sets from earlier releases are no longer supported, but may be in use in your current Oracle Database installation.
- Default Rule Sets
  Oracle Database Vault provides a set of default rule sets that you can customize for your needs.
- Creating a Rule Set
  To create a rule set, you first create the rule set itself, and then you can edit the rule set to associate it with one or more rules.
- Creating a Rule to Add to a Rule Set
  A rule defines the behavior that you want to control; a rule set is a named collection of rules.
- Modifying a Rule Set
  You can use the `DBMS_MACADM.UPDATE_RULE_SET` procedure to modify the definition of a rule set.
- Deleting a Rule Set
  Before you delete a rule set, you must remove any rules from the rule set.
- How Rule Sets Work
  Understanding how rule sets work helps to create more effective rule sets.
- Tutorial: Configuring Two-Person Integrity, or Dual Key Security
  This tutorial demonstrates how to use Oracle Database Vault to control the authorization of two users.
- Guidelines for Designing Rule Sets
  Oracle provides guidelines for designing rule sets.
- How Rule Sets Affect Performance
  The number and complexity of rules can slow database performance.
- Rule Set and Rule Related Reports and Data Dictionary Views
  Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing rule sets and rules.

5.1 What Are Rule Sets?

A rule set is a collection of one or more rules.

You can associate the rule set with a realm authorization, factor assignment, command rule, or secure application role.

The rule set evaluates to true or false based on the evaluation of each rule it contains and the evaluation type (*All True* or *Any True*). A rule within a rule set is a PL/SQL expression that evaluates to true or false. You can create a rule and add the rule to multiple rule sets.

You can use rule sets to accomplish the following activities:

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

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

Related Topics

- Rule Set and Rule Related Reports and Data Dictionary Views
  Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing rule sets and rules.
- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.

5.2 Rule Sets and Rules in a Multitenant Environment

You can create a rule set and its associated rules in a PDB or an application root.

A common realm must use a common rule set when the associated realm or command rule is evaluated by Database Vault. The common rule set and its rules can only be created in the application root. After the common rule set is created, it exists in every container that is associated with the root where the common rule set is created. The common rule set can only include common rules.

To configure a common rule set and its rules, you must be commonly granted the `DV_OWNER` or `DV_ADMIN` role.

Related Topics

- Command Rules in a Multitenant Environment
  You can create common and local command rules in either the CDB root or the application root.

5.3 Default Rules and Rule Sets from Releases Earlier Than Release 12.2

Many default rules and rule sets from earlier releases are no longer supported, but may be in use in your current Oracle Database installation.

If you use default rules and rule sets from releases earlier than Oracle Database release 12.2, Oracle Database does not remove them during an upgrade in case you have customized them for your own use. If you customized these rules and rule sets, or use these older default rule sets, Oracle recommends that you re-implement the customized rules and rule sets by using the `ALTER SYSTEM` and `ALTER SESSION` command rules, and then disable and drop the old rules and rule sets. If you have not customized these rules and rule sets, or otherwise use them, you should drop these earlier rules and rule sets because the same functionality is available in later default command rules.

> **Note:**
>
> See the release 12.2 version of *Oracle Database Vault Administrator's Guide* for a full listing of the rules and rule sets that may be affected.

5.4 Default Rule Sets

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

You can find a full list of rule sets by querying the `DBA_DV_RULE_SET` data dictionary view. To find rules that are associated with a rule set, query the `DBA_DV_RULE_SET_RULE` data dictionary view.

The default rule sets are as follows:

- `Allow Dumping Datafile Header` prevents the dumping of data blocks.
- `Allow Fine Grained Control for Alter System` enables you to control the ability of users to set initialization parameters using the `ALTER SYSTEM` SQL statement.
- `Allow Sessions` controls the ability to create a session in the database. This rule set enables you to add rules to control database logins using the `CONNECT` command rule. The `CONNECT` command rule is useful to control or limit `SYSDBA` access to programs that require its use. This rule set is not populated.
- `Can 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/Profiled` controls the roles that manage user accounts and profiles, through the `CREATE USER`, `DROP USER`, `CREATE PROFILE`, `ALTER PROFILE`, or `DROP PROFILE` statements.
- `Can Maintain Own Account` allows the accounts with the `DV_ACCTMGR` role to manage user accounts and profiles with the `ALTER USER` statement. Also allows individual accounts to change their own password using the `ALTER USER` statement. See DV_ACCTMGR Database Vault Account Manager Role for more information about the `DV_ACCTMGR` role.
- `Disabled` is a convenience rule set to quickly disable security configurations like realms, command rules, factors, and secure application roles.
- `Enabled` is a convenience rule set to quickly enable system features.

- `Not allow to set OPTIMIZER_SECURE_VIEW_MERGING to True` prevents the `OPTIMIZER_SECURE_VIEW_MERGING` initialization parameter from being set to `TRUE`.
- `Not allow to set OS_ROLES to True` prevents the `OS_ROLES` initialization parameter from being set to `TRUE`.
- `Not allow to set PLSQL_DEBUG to True` prevents the `PLSQL_DEBUG` initialization parameter from being set to `TRUE`.
- `Not allow to set REMOTE_OS_ROLES to True` prevents the `REMOTE_OS_ROLES` initialization parameter from being set to `TRUE`.
- `Not allow to set SQL92_SECURITY to False` prevents the `SQL92_SECURITY` from being set to `FALSE`.

5.5 Creating a Rule Set

To create a rule set, you first create the rule set itself, and then you can edit the rule set to associate it with one or more rules.

You can associate a new rule with the rule set, add existing rules to the rule set, or delete a rule association from the rule set.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Run the `DBMS_MACADM.CREATE_RULE_SET` statement to create the rule set.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'Evaluation failed',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
  handler          => '',
  is_static        => TRUE,
  scope            => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

   In this specification:

   - `rule_set_name` can be up to 128 characters in mixed-case. Spaces are allowed. Oracle suggests that you start the name with a verb and complete it with the realm or command rule name to which the rule set is attached.The `DBA_DV_RULE_SET` data dictionary view lists existing rule sets.
   - `description` can be 1024 characters in mixed-case. You may want to document the business requirement of the rule set (for example, `Rule set to limit access to SQL*Plus`).

- `enabled` controls whether the rule set is enabled or disabled. `DBMS_MACUTL.G_YES` enables the rule set; `DBMS_MACUTL.G_NO` disables it. The default is `DBMS_MACUTL.G_YES`.
- `eval_options` is used if you plan to have multiple rules associated with the rule set. `DBMS_MACUTL.G_RULESET_EVAL_ALL` means all rules must evaluate to `TRUE`; `DBMS_MACUTL.G_RULESET_EVAL_ANY` means at least one rule must evaluate to `TRUE`.
- `audit_options` applies only to traditional auditing, not unified auditing environments. Starting with Oracle Database release 23c, traditional auditing is desupported. Only the `DBMS_MACUTL.G_RULESET_AUDIT_OFF` setting is available for the `audit_options` parameter. To audit a rule set, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a rule set.
- `fail_options` designates whether to show (`DBMS_MACUTL.G_RULESET_FAIL_SHOW`) to not show (`DBMS_MACUTL.G_RULESET_FAIL_SILENT`) error messages. An advantage of selecting `DBMS_MACUTL.G_RULESET_FAIL_SILENT` 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 they do not see any error messages.
- `fail_message` is a text string error message up to 80 characters in mixed-case, to associate with the fail code you specify for `fail_code`. If you do not specify an error message, then Oracle Database Vault displays a generic error message.
- `fail_code` is a number in the range of `-20000` to `-20999` or `20000` to `20999` to associate with the `fail_message` parameter. If you omit this setting, then Oracle Database Vault displays a generic error code.
- `handler_options` enables you to include handler code to define custom event handler logic. `DBMS_MACUTL.G_RULESET_HANDLER_OFF` disables error handling (default), `DBMS_MACUTL.G_RULESET_HANDLER_FAIL` calls handler on rule set failure, and `DBMS_MACUTL.G_RULESET_HANDLER_SUCCESS` calls handler on rule set success.
- `handler` is a PL/SQL function or procedure that defines the custom event handler logic. 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.
  Write the expression as a fully qualified procedure (such as *`schema.procedure_name`*). Do not include any other form of SQL statements. If you are using application package procedures or standalone procedures, you must provide `DVSYS` with the `EXECUTE` privilege on the object. The procedure signature can be in one of the following two forms:
  - `PROCEDURE` *`my_ruleset_handler(p_ruleset_name`* `IN VARCHAR2,` *`p_ruleset_rules`* `IN BOOLEAN)`: Use this form when the name of the rule set and its return value are required in the handler processing.
  - `PROCEDURE` *`my_ruleset_handler`*: Use this form when the name of the rule set and its return value are not required in the handler processing.

  Be aware that you cannot use invoker's rights procedures as event handlers. Doing so can cause the rule set evaluation to fail unexpectedly. Only use definer's rights procedures as event handlers.

  Use the following syntax:

  `myschema.my_ruleset_handler`

- `is_static` determines how often a rule set is evaluated when it is accessed. `TRUE` evaluates the rule set once during the user session. After that, the value is re-used. `FALSE` evaluates the rule set each time the rule set is called. The default is `FALSE`.
- `scope` defines whether the rule set is created in a PDB (`DBMS_MACUTL.G_SCOPE_LOCAL`) or in an application root (`DBMS_MACUTL.G_SCOPE_COMMON`). If you create the common rule set in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example:

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

At this stage the rule set creation is complete.

3. Optionally, add one or more rules to the rule set.

   The `DBA_DV_RULE` data dictionary view lists existing rules.

   For example:

   ```
   BEGIN
    DBMS_MACADM.ADD_RULE_TO_RULE_SET(
     rule_set_name => 'Limit_DBA_Access',
     rule_name     => 'Is Database Administrator',
     rule_order    => 1,
     enabled       => DBMS_MACUTL.G_YES);
   END;
   /
   ```

Related Topics

- Creating a Rule
  You can create a new rule or use the default Oracle Database Vault rules.
- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.
- Oracle Database Vault PL/SQL Rule Set Functions
  Oracle Database Vault provides functions to use in rule sets to inspect the SQL statement that the rule set protects.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

5.6 Creating a Rule to Add to a Rule Set

A rule defines the behavior that you want to control; a rule set is a named collection of rules.

- What Are Rules?
  A rule is an expression that checks if a particular condition is true or false.

- Default Rules
  Default rules are rules that have commonly used behavior, such as checking if an action evaluates to true or false.
- Creating a Rule
  You can create a new rule or use the default Oracle Database Vault rules.
- Adding Existing Rules to a Rule Set
  After you have created one or more rules, you can add them to a rule set.
- Modifying a Rule Set
  You can use the `DBMS_MACADM.UPDATE_RULE` procedure to modify the definition of a rule.
- Removing a Rule from a Rule Set
  Before you remove a rule from a rule set, you must remove references to it from rule sets.

5.6.1 What Are Rules?

A rule is an expression that checks if a particular condition is true or false.

This expression enables Oracle Database Vault to perform an action based on the result of the rule evaluation during runtime. For example, the `Is Database Administrator` default rule checks if the session user is `SYS` or has the `DBA` role enabled or granted. You can create an Oracle Database Vault rule set that includes the rule to prevent users who fail this rule check from accessing a realm that should only be accessed by users with the `DBA` role. In addition, you can create Oracle Database Vault command rules that prevent users who fail a rule check from executing critical SQL commands that should only be used by users with the `DBA` role or `SYS`.

The expression that you use to define the rule must be a PL/SQL Boolean expression. You can use standard Oracle Database functions to build the expression, such as the `SYS_CONTEXT` function. An example of a rule expression using the `SYS_CONTEXT` function is as follows:

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

This rule expression translates to "Check if the currently logged in user is user `RLAYTON`."

When you create a rule, you can define a scope for it, that is, whether the rule is created in a PDB or in an application root.

You can create rules during the rule set creation process, or independently of it. After you create the rule, you can associate a rule set with one or more additional rules.

If you create a new rule during the rule set creation process, the rule is automatically added to the current rule set. You also can add existing rules to the rule set. Alternatively, you can omit adding rules to the rule set and use it as a template for rule sets you may want to create in the future.

You can add as many rules that you want to a rule set, but for better design and performance, you should keep the rule sets simple. Oracle provides guidelines for designing rule sets.

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, then Oracle Database Vault evaluates the rule set to true without evaluating its rules.

Related Topics

- How Rule Sets Work
  Understanding how rule sets work helps to create more effective rule sets.
- Guidelines for Designing Rule Sets
  Oracle provides guidelines for designing rule sets.

5.6.2 Default Rules

Default rules are rules that have commonly used behavior, such as checking if an action evaluates to true or false.

You can find a full list of rules by querying the `DBA_DV_RULE` data dictionary view. The following table lists the current default Oracle Database rules.

Table 5-1 Current Default Oracle Database Vault Rules

Rule	Description
Are Dest Parameters Allowed	Checks if the current SQL statement attempts to alter initialization parameters related to the size limit of a dump
Are Dump Parameters Allowed	Checks if the current SQL statement attempts to alter initialization parameters related to the destination of a dump
False	Evaluates to `FALSE`
Is Alter DVSYS Allowed	Note: This default rule has been deprecated. Checks if the logged-in user can run the `ALTER USER` statement on other users successfully
Is Database Administrator	Checks if a user has been granted the `DBA` role
Is Drop User Allowed	Checks if the logged in user can drop users
Is Dump of Block Allowed	Checks if the dumping of blocks is allowed
Is First Day of Month	Checks if the specified date is the first day of the month
Is Label Administrator	Checks if the user has been granted the `LBAC_DBA` role
Is Last Day of Month	Checks if the specified date is the last day of the month
Is Parameter Value False	Checks if a specified parameter value has been set to `FALSE`
Is Parameter Value None	Checks if a specified parameter value has been set to `NONE`
Is Parameter Value Not False	Checks if a specified parameter value has been set to `<> FALSE`
Is Parameter Value Not None	Checks if a specified parameter value has been set to `<> NONE`
Is Parameter Value Not Off	Checks if a specified parameter value has been set to `<> OFF`
Is Parameter Value Not On	Checks if a specified parameter value has been set to `<> ON`
Is Parameter Value Not True	Checks if a specified parameter value has been set to `<> TRUE`

Table 5-1 (Cont.) Current Default Oracle Database Vault Rules

Rule	Description
Is Parameter Value Off	Checks if a specified parameter value has been set to `OFF`
Is Parameter Value On	Checks if a specified parameter value has been set to `ON`
Is Parameter Value True	Checks if a specified parameter value has been set to `TRUE`
Is SYS or SYSTEM User	Checks if the user is `SYS` or `SYSTEM`
Is Security Administrator	Checks if a user has been granted the `DV_ADMIN` role
Is Security Owner	Checks if a user has been granted the `DV_OWNER` role
Is User Manager	Checks if a user has been granted the `DV_ACCTMGR` role
Login User Is Object User	Checks if the logged in user is the same as the user about to be altered by the current SQL statement
No Exempt Access Policy Role	Checks if the user has been granted the `EXEMPT ACCESS POLICY` role or is user `SYS`
Not Export Session	Obsolete
True	Evaluates to `TRUE`

5.6.3 Creating a Rule

You can create a new rule or use the default Oracle Database Vault rules.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Run the `DBMS_MACADM.CREATE_RULE` statement to create the rule.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Is SYSADM Administrator',
  rule_expr  =>'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SYSADM''',
  scope      => DBMS_MACUTL.G_SCOPE_LOCAL,
  is_static  => TRUE);
END;
/
```

   In this specification:

   - `rule_name` up to 90 characters in mixed-case. Spaces are allowed. The `DBA_DV_RULE` data dictionary view lists existing rules. The `DBA_DV_RULE_SET_RULE` lists rule sets that are associated with rules. 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` parameter, make the name explicit but be sure to not exceed over 90 characters.

- `rule_expr` is a 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'''
  ```

  Enter a PL/SQL expression that fits the following requirements:

  - It is valid in a SQL `WHERE` clause.
  - It can be a freestanding and valid PL/SQL Boolean expression such as the following:

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

  - It must evaluate to a Boolean (`TRUE` or `FALSE`) value.
  - It must be no more than 1024 characters long.
  - It can contain existing and compiled PL/SQL functions from the current database instance. Ensure that these are fully qualified functions (such as *`schema`*.*`function_name`*). Do not include any other form of SQL statements.

    Be aware that you cannot use invoker's rights procedures with rule expressions. Doing so will cause the rule evaluation to fail unexpectedly. Only use definer's rights procedures with rule expressions.

    If you want to use application package functions or standalone functions, you must grant the `DVSYS` account the `EXECUTE` privilege on the function. Doing so reduces the chances of errors when you add new rules.
  - Ensure that the rule works. You can test the syntax by running the following statement in SQL*Plus:

    ```
    SELECT rule_expression FROM DUAL;
    ```

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

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

    You could test this expression as follows:

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

    For the Boolean example listed earlier, you would enter the following:

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

- `scope` defines whether the rule is created in a PDB (`DBMS_MACUTL.G_SCOPE_LOCAL`) or in an application root (`DBMS_MACUTL.G_SCOPE_COMMON`).
- `is_static` specifies whether the evaluation of a rule should be static or dynamic. When a rule is specified as static, then the result of the first rule evaluation in a session will be cached and used whenever the rule needs to be evaluated again in the session. The default is `FALSE`.

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

Related Topics

- Adding Existing Rules to a Rule Set
  After you have created one or more rules, you can add them to a rule set.
- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.
- Oracle Database Vault PL/SQL Rule Set Functions
  Oracle Database Vault provides functions to use in rule sets to inspect the SQL statement that the rule set protects.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

5.6.4 Adding Existing Rules to a Rule Set

After you have created one or more rules, you can add them to a rule set.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_RULE` data dictionary view to find the rule to add to a rule set.

   ```
   SELECT NAME FROM DBA_DV_RULE
   ORDER BY NAME;
   ```

3. Query the `DBA_DV_RULE_SET` data dictionary view to find the rule set to which you want to add the rule.

   ```
   SELECT RULE_SET_NAME
   FROM DBA_DV_RULE_SET
   ORDER BY RULE_SET_NAME;
   ```

   You can also query the `DBA_DV_RULE_SET_RULE` data dictionary view to find if the rule has already been associated with a rule set.

4. Run the `DBMS_MACADM.ADD_RULE_TO_RULE_SET` to add the rule to a rule set.

   For example:

   ```
   BEGIN
    DBMS_MACADM.ADD_RULE_TO_RULE_SET(
     rule_set_name => 'Limit_DBA_Access',
     rule_name     => 'Is SYSADM Administrator',
     rule_order    => 1,
     enabled       => DBMS_MACUTL.G_NO,
     scope         => );
   END;
   /
   ```

   In this specification:

   - `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` determines whether the rule should be checked when the rule set is evaluated. `DBMS_MACUTL.G_YES` (default). Enables the rule to be checked during the rule set evaluation. `DBMS_MACUTL.G_NO` Prevents the rule from being checked during the rule set evaluation.

- `scope` defines whether the rule is created in a PDB (`DBMS_MACUTL.G_SCOPE_LOCAL`) or in an application root (`DBMS_MACUTL.G_SCOPE_COMMON`).

Related Topics

- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.

5.6.5 Modifying a Rule Set

You can use the `DBMS_MACADM.UPDATE_RULE` procedure to modify the definition of a rule.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Find the rule and check its definition.

   For example:

   ```
   SELECT * FROM DBA_DV_RULE ORDER BY NAME;
   ```

3. Run the `DBMS_MACADM.UPDATE_RULE` statement.

   For example:

   ```
   BEGIN
    DBMS_MACADM.UPDATE_RULE(
     rule_name  => 'Check UPDATE operations',
     rule_expr  =>'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SYSADM'' AND
                   (
                     UPPER(SYS_CONTEXT(''USERENV'',''MODULE'')) LIKE
   ''APPSRVR%'' OR
                     UPPER(SYS_CONTEXT(''USERENV'',''MODULE'')) LIKE
   ''DBAPP%'' )'
                  );
   END;
   /
   ```

Related Topics

- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.

5.6.6 Removing a Rule from a Rule Set

Before you remove a rule from a rule set, you must remove references to it from rule sets.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_RULE` data dictionary view to find the rule that you want to remove from a rule set.

   ```
   SELECT NAME FROM DBA_DV_RULE
   ORDER BY NAME;
   ```

3. Query the `DBA_DV_RULE_SET_RULE` data dictionary views to find rule sets that are associated with the rule.

   For example:

   ```
   SELECT RULE_SET_NAME
   FROM DBA_DV_RULE_SET_RULE
   WHERE RULE_NAME = 'Is SYSADM Administrator';
   ```

4. Execute the `DBMS_MACADM.DELETE_RULE_FROM_RULE_SET` procedure to remove the rule from the rule set.

   For example:

   ```
   BEGIN
    DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
     rule_set_name => 'Limit_DBA_Access',
     rule_name     => 'Is SYSADM Administrator');
   END;
   /
   ```

After you remove the rule from the rule set, the rule still exists. If you want, you can associate it with other rule sets. You can also delete the rule by executing the `DBMS_MACADM.DELETE_RULE`. For example:

```
EXEC DBMS_MACADM.DELETE_RULE('Is SYSADM Administrator');
```

Related Topics

- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.

5.7 Modifying a Rule Set

You can use the `DBMS_MACADM.UPDATE_RULE_SET` procedure to modify the definition of a rule set.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Find the rule set and check its definition.

   For example:

   ```
   SELECT * FROM DBA_DV_RULE_SET ORDER BY RULE_SET_NAME;
   ```

3. Run the `DBMS_MACADM.UPDATE_RULE_SET` statement.

   For example:

   ```
   BEGIN
    DBMS_MACADM.UPDATE_RULE_SET(
     rule_set_name    => 'Limit_DBA_Access',
     description      => 'DBA access through predefined processes',
     enabled          => DBMS_MACUTL.G_NO,
     eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
     audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
     fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
     fail_message     => 'Access denied!',
     fail_code        => 20900,
     handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
     handler          => '',
   ```

```
  is_static        =  TRUE);
END;
/
```

After you update a rule set, if it has traditional auditing set (`audit_options`), then this auditing disabled because traditional auditing is desupported starting in release 23c. Only the `DBMS_MACUTL.G_RULESET_AUDIT_OFF` setting is available for the `audit_options` parameter. To audit a rule set, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a rule set.

Related Topics

- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

5.8 Deleting a Rule Set

Before you delete a rule set, you must remove any rules from the rule set.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_RULE_SET` data dictionary view to find the rule set that you want to delete.

   ```
   SELECT RULE_SET_NAME
   FROM DBA_DV_RULE_SET
   ORDER BY RULE_SET_NAME;
   ```

3. Query the `DBA_DV_RULE_SET_RULE` data dictionary view to ensure that no rules are associated with the rule set that you want to delete.

   For example:

   ```
   SELECT RULE_NAME
   FROM DBA_DV_RULE_SET_RULE
   WHERE RULE_SET_NAME = 'Limit_DBA_Access';
   ```

4. If necessary, run `DBMS_MACADM.DELETE_RULE_FROM_RULE_SET` remove the rules that are associated with the rule set.

   For example:

   ```
   BEGIN
    DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
     rule_set_name => 'Limit_DBA_Access',
     rule_name     => 'Is SYSADM Administrator');
   END;
   /
   ```

5. Run the `DBMS_MACADM.DELETE_RULE_SET` procedure to delete the rule set.

   For example:

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

Related Topics

- Oracle Database Vault Rule Set APIs
  You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.

5.9 How Rule Sets Work

Understanding how rule sets work helps to create more effective rule sets.

- How Oracle Database Vault Evaluates Rules
  Oracle Database Vault evaluates the rules within a rule set as a collection of expressions.
- Nested Rules within a Rule Set
  You can nest one or more rules within the rule set.
- Creating Rules to Apply to Everyone Except One User
  You can also create rules to apply to everyone *except* one user (for example, a privileged user).

5.9.1 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 the `eval_options` parameter in the `DBMS_MACADM.CREATE_RULE_SET` or `DBMS_MACADM.UPDATE_RULE_SET` procedure to `DBMS_MACUTL.G_RULESET_EVAL_ALL` and if a rule evaluates to false, then the evaluation stops at that point, instead of attempting to evaluate the rest of the rules in the rule set. Similarly, if `eval_options` is set to `DBMS_MACUTL.G_RULESET_EVAL_ANY` and if a rule evaluates to true, the evaluation stops at that point. If a rule set is disabled, then Oracle Database Vault evaluates it to true without evaluating its rules.

5.9.2 Nested Rules within a Rule Set

You can nest one or more rules within the rule set.

For example, suppose you want to create a nested rule, `Is Corporate Network During Maintenance`, that performs the following two tasks:

- It limits table modifications only when the database session originates within the corporate network.
- It restricts table modifications during the system maintenance window scheduled between 10:00 p.m. and 10:59 p.m.

The rule definition would be as follows:

```
DVF.F$NETWORK = 'Corporate' AND TO_CHAR(SYSDATE,'HH24') between '22' AND '23'
```

Related Topics

- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the `DVF` schema functions when you use the `DBMS_MACADM` PL/SQL package to manage the various factors.
- Configuring Factors
  Factors allow you to create and use complex attributes through PL/SQL to make Oracle Database Vault authorization decisions.

5.9.3 Creating Rules to Apply to Everyone Except One User

You can also create rules to apply to everyone *except* one user (for example, a privileged user).

- To create a rule that excludes specific users, user the `SYS_CONTEXT` function.

For example:

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

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

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

This tutorial demonstrates how to use Oracle Database Vault to control the authorization of two users.

- About This Tutorial
  In this tutorial, you configure a rule set that defines two-person integrity (TPI).
- Step 1: Create Users for This Tutorial
  You must create two users for this tutorial, `patch_boss` and `patch_user`.
- Step 2: Create a Function to Check if User patch_boss Is Logged In
  The behavior of the Database Vault settings will be determined by the function.
- Step 3: Create Rules, a Rule Set, and a Command Rule to Control User Access
  Next, you must create two rules, a rule set to which you will add them, and a command rule.
- Step 4: Test the Users' Access
  After the rules have been created, they are ready to be tested.
- Step 5: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

5.10.1 About This Tutorial

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

This feature is also called dual key security, dual key connection, and two-man rule security. In this type of security, two users are required to authorize an action instead of one user.

The idea is that one user provides a safety check for the other user before that user can proceed with a task. Two-person integrity provides an additional layer of security for actions that potentially can be dangerous. This type of scenario is often used for tasks such as database patch updates, which is what this tutorial will demonstrate. One user, `patch_user` must log in to perform a database patch upgrade, but the only way that they can do this is if their 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.

5.10.2 Step 1: Create Users for This Tutorial

You must create two users for this tutorial, `patch_boss` and `patch_user`.

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

To create the users:

1. Log in to a PDB as a user who has been granted the `DV_ACCTMGR` role.
2. Create the following users and grant them the `CREATE SESSION` privilege.

   ```
   GRANT CREATE SESSION TO patch_boss IDENTIFIED BY password;
   GRANT CREATE SESSION TO patch_user IDENTIFIED BY password;
   ```

   Replace *password* with a password that meets the password complexity requirements of the user's profile.
3. Connect as user `SYS` with the `SYSDBA` administrative privilege.
4. Grant the following privileges to the `DV_OWNER` or `DV_ADMIN` user.

   For example:

   ```
   GRANT CREATE PROCEDURE TO sec_admin_owen;
   GRANT SELECT ON V_$SESSION TO sec_admin_owen;
   ```

   The `V_$SESSION` table is the underlying table for the `V$SESSION` dynamic view.

In a real-world scenario, you also would log in as the `DV_OWNER` user and grant the `DV_PATCH_ADMIN` role to user `patch_user` (but not to `patch_boss`). But because you are not really going to perform a database patch upgrade in this tutorial, you do not need to grant this role to user `patch_user`.

Related Topics

- *Oracle Database Security Guide*

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

The behavior of the Database Vault settings will be determined by the function.

The function that you must create, `check_boss_logged_in`, does just that: When user `patch_user` tries to log in to the database instance, it checks if user `patch_boss` is already logged in by querying the `V$SESSION` data dictionary view.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Create the `check_boss_logged_in` function as follows:

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

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

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

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

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

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

5.10.4 Step 3: Create Rules, a Rule Set, and a Command Rule to Control User Access

Next, you must create two rules, a rule set to which you will add them, and a command rule.

The rule set triggers the `check_boss_logged_in` function when user `patch_user` tries to logs in to the database.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Create the `Check if Boss Is Logged In` rule, which checks that the `patch_user` user is logged in to the database. In the definition, replace `sec_admin_owen` with the name of the `DVOWNER` or `DV_ADMIN` user who created the `check_boss_logged_in` function.

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

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

   Enter the user name, `PATCH_USER`, in upper-case letters, which is how the `SESSION_USER` parameter stores it.
3. Create the `Allow Connect for Other Database Users` rule, which ensures that the user logged in (`patch_user`) is not user `patch_boss`. It also enables all other valid users to log in.

   ```
   BEGIN
    DBMS_MACADM.CREATE_RULE(
   ```

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

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

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

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

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

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

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

5.10.5 Step 4: Test the Users' Access

After the rules have been created, they are ready to be tested.

1. Exit SQL*Plus.

```
EXIT
```

2. Create a second shell, for example:

```
xterm &
```

3. In the first shell, try to log in as user `patch_user`.

```
sqlplus patch_user@pdb_name
Enter password: password

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

Enter user-name:
```

User `patch_user` cannot log in until user `patch_boss` is already logged in. (Do not try the `Enter user-name` prompt yet.)

4. In the second shell and then log in as user `patch_boss`.

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

User `patch_boss` can log in.

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

```
Enter user_name: patch_user
Enter password: password
```

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

5.10.6 Step 5: Remove the Components for This Tutorial

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

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

```
EXIT
```

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

```
DROP USER patch_boss;
DROP USER patch_user;
```

3. Connect as a user `SYS` with the `SYSDBA` administrative privilege and revoke the privileges that you had granted to the `DV_OWNER` or `DV_ADMIN` user.

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

4. Connect as the `DV_OWNER` or `DV_ADMIN` user and drop the rules, rule set, and command rule, in the order shown.

```
DROP FUNCTION check_boss_logged_in;
EXEC DBMS_MACADM.DELETE_COMMAND_RULE('CONNECT', '%', '%');
EXEC DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('Dual Connect for Boss and
Patch', 'Check if Boss Is Logged In');
EXEC DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('Dual Connect for Boss and
Patch', 'Allow Connect for Other Database Users');
EXEC DBMS_MACADM.DELETE_RULE('Check if Boss Is Logged In');
EXEC DBMS_MACADM.DELETE_RULE('Allow Connect for Other Database Users');
```

```
EXEC DBMS_MACADM.DELETE_RULE_SET('Dual Connect for Boss and Patch');
COMMIT;
```

5.11 Guidelines for Designing Rule Sets

Oracle provides guidelines for designing rule sets.

- You can share rules among multiple rule sets. This lets you develop a library of reusable rule expressions. Oracle recommends that you design such rules to be discrete, single-purpose expressions.
- You can design a rule set so that its evaluation is static, that is, it is evaluated only once during a user session. Alternatively, it can be evaluated each time the rule set is accessed. If the rule set is evaluated only once, then the evaluated value is reused throughout the user session each time the rule set is accessed. Using static evaluation is useful in cases where the rule set must be accessed multiple times but the conditions on which the rule set depend do not change during that session. An example would be a `SELECT` command rule associated with a rule set when the same `SELECT` statement occurs multiple times and if the evaluated value is acceptable to use again, rather than evaluating the rule set each time the `SELECT` occurs.

  To control the static evaluation of the rule set, set the `is_static` parameter of the `CREATE_RULE_SET` or `UPDATE_RULE_SET` procedures of the `DBMS_MACADM` PL/SQL package. See DBMS_MACADM Rule Set Procedures for more information.
- Use Oracle Database Vault factors in your rule expressions to provide reusability and trust in the values used by your rule expressions. Factors can provide contextual information to use in your rules expressions.
- You can use custom event handlers to extend Oracle Database Vault security policies to integrate external systems for error handling or alerting. Using Oracle utility packages such as `UTL_TCP`, `UTL_HTTP`, `UTL_MAIL`, `UTL_SMTP`, or `DBMS_AQ` can help you to achieve this type of integration.
- Test rule sets thoroughly for various accounts and scenarios either on a test database or on a test realm or command rule for nonsensitive data before you apply them to realms and command rules that protect sensitive data. You can test rule expressions directly with the following SQL statement:

  ```
  SQL> SELECT SYSDATE from DUAL where rule expression
  ```
- You can nest rule expressions inside a single rule. This helps to achieve more complex situations where you would need a logical `AND` for a subset of rules and a logical `OR` with the rest of the rules. For example, suppose you want to create a nested rule that performs the following two tasks:
  - Limits table modifications only when the database session originates within the corporate network
  - Restricts table modifications during the system maintenance window scheduled between 10:00 p.m. and 10:59 p.m.

  A rule definition for this scenario could be as follows:

  ```
  DVF.F$NETWORK = 'Corporate' AND TO_CHAR(SYSDATE,'HH24') between '22' AND '23'
  ```
- You cannot use invoker's rights procedures with rule expressions. Only use definer's rights procedures with rule expressions.

5.12 How Rule Sets Affect Performance

The number and complexity of rules can slow database performance.

Rule sets govern the performance for execution of certain operations. For example, if you have a very large number of rules in a rule set governing a `SELECT` statement, performance could degrade significantly.

If you have rule sets that require many rules, performance improves if you move all the rules to logic defined in a single PL/SQL standalone or package function. However, if a rule is used by other rule sets, there is little performance effect on your system.

If possible, consider setting the rule set to use static evaluation, assuming this is compatible with the associated command rule's usage. See Guidelines for Designing Rule Sets for more information.

You can check system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Cloud Control, which is installed by default with Oracle Database), Automatic Workload Repository (AWR), and `TKPROF`.

Related Topics

- *Oracle Database Performance Tuning Guide*
- *Oracle Database SQL Tuning Guide*

5.13 Rule Set and Rule Related Reports and Data Dictionary Views

Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing rule sets and rules.

Table 5-2 lists the Oracle Database Vault reports.

Table 5-2 Oracle Enterprise Manager Reports Related to Rule Sets

Report	Description
Rule Set Configuration Issues Report	Lists rule sets that have no rules defined or enabled
Secure Application Configuration Issues Report	Lists secure application roles that have incomplete or disabled rule sets
Command Rule Configuration Issues Report	Lists rule sets that are incomplete or disabled

Table 5-3 lists data dictionary views that provide information about existing rules and rule sets.

Table 5-3 Data Dictionary Views Used for Rules and Rule Sets

Data Dictionary View	Description
`DBA_DV_RULE`	Lists the rules that have been defined

Table 5-3 (Cont.) Data Dictionary Views Used for Rules and Rule Sets

Data Dictionary View	Description
DBA_DV_RULE_SET	Lists the rule sets that have been created
DBA_DV_RULE_SET_RULE	Lists rules that are associated with existing rule sets

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

6
Configuring Command Rules

You can create command rules or use the default command rules to protect DDL and DML statements.

- What Are Command Rules?
  A command rule applies Oracle Database Vault protections with an Oracle Database SQL statement, such as `ALTER SESSION`.
- Default Command Rules
  Oracle Database Vault provides default command rules, based on commonly used SQL statements.
- SQL Statements That Can Be Protected by Command Rules
  You can protect a large number of SQL statements by using command rules.
- Creating a Command Rule
  You can create a different types of command rules using different command rule APIs.
- Modifying a Command Rule
  You can use the `DBMS_MACADM.UPDATE_COMMAND_RULE`, `DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE`, `DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE`, and `DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE` procedures to modify the definition of a command rule.
- Deleting a Command Rule
  Before you delete a command rule, you can locate the various references to it by querying the command rule-related Oracle Database Vault views.
- How Command Rules Work
  Command rules follow a set of steps to check their associated components.
- Tutorial: Using a Command Rule to Control Table Creations by a User
  In this tutorial, you create a simple local command rule to control whether users can create tables in the `SCOTT` schema.
- Guidelines for Designing Command Rules
  Oracle provides guidelines for designing command rules.
- How Command Rules Affect Performance
  The performance of a command rule depends on the complexity of the rules in the rule set associated with the command rule.
- Command Rule Related Reports and Data Dictionary View
  Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing command rules.

6.1 What Are Command Rules?

A command rule applies Oracle Database Vault protections with an Oracle Database SQL statement, such as `ALTER SESSION`.

- About Command Rules
  A command rule protects Oracle Database SQL statements that affect one or more database objects.
- Command Rules in a Multitenant Environment
  You can create common and local command rules in either the CDB root or the application root.
- Types of Command Rules
  In addition to command rules for many SQL statements, you can create command rules specifically for the `CONNECT`, `ALTER SYSTEM`, and `ALTER SESSION` SQL statements.

6.1.1 About Command Rules

A command rule protects Oracle Database SQL statements that affect one or more database objects.

These statements can include `SELECT`, `ALTER SYSTEM`, database definition language (DDL), and data manipulation language (DML) statements.

To customize and enforce the command rule, you associate it with a rule set, which is a collection of one or more rules. The command rule is enforced at run time. Command rules affect anyone who tries to use the SQL statements it protects, regardless of the realm in which the object exists.

You can use command rules to protect a wide range of SQL statements, in addition to basic Oracle Database DDL and DML statements. For example, you can protect statements that are used with Oracle Flashback Technology.

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

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

Command rules can be categorized as follows:

- **Command rules that have a system-wide scope.** With this type, in most cases, you can only create one command rule for each database instance.
- **Command rules that are schema specific.** An example of a schema-specific command rule is a command rule for the `DROP TABLE` statement. You can create only one CONNECT command rule for each schema.
- **Command rules that are object specific.** An example is creating a command rule for the `DROP TABLE` statement with a specific table included in the command rule definition.

When a user runs a statement affected by a command rule, Oracle Database Vault checks the realm authorization first. If it finds no realm violation and if the associated command rules are enabled, then Database Vault evaluates the associated rule sets. If all the rule sets evaluate to `TRUE`, then the statement is authorized for further

processing. If any of the rule sets evaluate to FALSE, then the statement is not allowed to be run and a command rule violation is raised.

You can define a command rule that uses factors for the CONNECT event to permit or deny sessions after the usual steps–user authentication process, factor initialization, and Oracle Label Security integration–are complete. For example, you can configure a command rule that allows DDL statements such as CREATE TABLE, DROP TABLE, and ALTER TABLE in the BIZAPP schema to be authorized after business hours, but not during business hours.

You can run reports on the command rules that you create in Oracle Database Vault.

You cannot create command rules that block SYS from executing SYS-owned procedures.

Related Topics

- Oracle Database Vault Command Rule APIs
  The DBMS_MACADM PL/SQL package provides procedures for configuring command rules. .
- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.
- SQL Statements That Can Be Protected by Command Rules
  You can protect a large number of SQL statements by using command rules.

6.1.2 Command Rules in a Multitenant Environment

You can create common and local command rules in either the CDB root or the application root.

Common command rules can be associated only with common realms, rule sets, and rules. Local command rules can be associated only with local realm, rule sets, and rules.

To apply these command rules to the entire multitenant environment, you must run the command rule procedures from the CDB root or application root as a common user who has been granted the DVADM or DVOWNER role. A common command rule that is created in the CDB root will be applied to all PDBs in that CDB environment. A common command rule that is created in the application root will only be applied to the PDBs that are associated with this application root. To propagate the command rule to the PDBs that are associated with the CDB root or application root, you must synchronize the PDB. For example, to synchronize an application root called saas_sales_app to its application PDBs:

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

To synchronize a common command rule in the CDB root to a PDB:

```
ALTER PLUGGABLE DATABASE APPLICATION APP$CDB$SYSTEM SYNC;
```

You can check a user's roles by querying the USER_ROLE_PRIVS data dictionary view. To find information about command rules, query the DBA_DV_COMMAND_RULE data dictionary view.

6.1.3 Types of Command Rules

In addition to command rules for many SQL statements, you can create command rules specifically for the CONNECT, ALTER SYSTEM, and ALTER SESSION SQL statements.

- CONNECT Command Rule
  The `DBMS_MACADM.CREATE_CONNECT_CMD_RULE` procedure creates a user-specific `CONNECT` command rule.
- ALTER SESSION and ALTER SYSTEM Command Rules
  You can create different kinds of `ALTER SESSION` and `ALTER SYSTEM` command rules that provide fine-grained control for these SQL statements.

6.1.3.1 CONNECT Command Rule

The `DBMS_MACADM.CREATE_CONNECT_CMD_RULE` procedure creates a user-specific `CONNECT` command rule.

This type of command rule specifies a user, an associated rule set, an enablement status, and where to run the `CONNECT` command rule. You can enable or disable the `CONNECT` command rule, or you can set it to use simulation mode. In simulation mode, violations to the command rule are logged in a designated log table with sufficient information to describe the error, such as the user name or SQL statement used.

You can create the `CONNECT` command rule in either the application root or in a specific PDB. The associated rule set must be consistent with the `CONNECT` command rule: if the `CONNECT` command rule is in the application root, then the rule set and rules must also be in the application root. You run the `CONNECT` command rule procedures from the CDB root as a common user. If the `CONNECT` command rule is local to a pluggable database (PDB), then you must run the `CONNECT` command rule creation command in that PDB, and the rule set and rules must be local.

The following example shows a `CONNECT` command rule definition that creates a local, enabled `CONNECT` command rule for the `HR` user. The rule set that is associated with this command rule is local to the current PDB.

```
BEGIN
DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
 rule_set_name   => 'Enabled',
 user_name       => 'HR',
 enabled         => DBMS_MACUTL.G_YES,
 scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

Related Topics

- CREATE_COMMAND_RULE Procedure
  The `CREATE_COMMAND_RULE` procedure creates both command and local command rules, which can be added to a rule set.
- Using Simulation Mode for Logging Realm and Command Rule Activities
  Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

6.1.3.2 ALTER SESSION and ALTER SYSTEM Command Rules

You can create different kinds of `ALTER SESSION` and `ALTER SYSTEM` command rules that provide fine-grained control for these SQL statements.

The procedures to create these types of command rules are as follows:

- `DBMS_MACADM.CREATE_COMMAND_RULE` creates `ALTER SESSION` and `ALTER SYSTEM` command rules that use clauses from the corresponding SQL statement, such as `ADVISE`, `CLOSE DATABASE LINK`, `COMMIT IN PROCEDURE`, and `SET` for `ALTER SESSION`, or `ARCHIVE_LOG`, `CHECK DATAFILES`, `CHECKPOINT`, and `SET` for `ALTER SYSTEM`.
- `DBMS_MACADM.CREATE_SESSION_EVENT` creates a command rule that is specific to the `ALTER SESSION SET EVENTS` SQL statement
- `DBMS_MACADM_CREATE_SYSTEM_EVENT` creates a command rule that is specific to the `ALTER SYSTEM SET EVENTS` SQL statement.

To create these command rules, you use the appropriate Database Vault procedure to specify the clause and if applicable, the parameter of the clause, in the creation statement. If the `ALTER SESSION` or `ALTER SYSTEM` command rule use the `SET EVENTS` setting, then you can use special parameters to specify events, components, and actions.

For example, for an `ALTER SYSTEM` command rule, you could specify the `SECURITY` clause and its `RESTRICTED SESSION` parameter from the `ALTER SYSTEM` SQL statement. To specify whether `RESTRICTED SESSION` is `TRUE` or `FALSE`, you must create a Database Vault rule and rule set, which can test for the validity of this sequence number.

To understand how this concept works, first create the following rule and rule set, which are designed to check if the `RESTRICTED SESSION` parameter is set to `TRUE`:

```
EXEC DBMS_MACADM.CREATE_RULE('RESTRICTED SESSION TRUE', 'UPPER(PARAMETER_VALUE) =
''TRUE''');

BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Check RESTRICTED SESSION for TRUE',
  description      => 'Checks if restricted session is true',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
  fail_message     => 'RESTRICTED SESSION is not TRUE',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler          => '',
  is_static        => false);
END;
/
EXEC DBMS_MACADM.ADD_RULE_TO_RULE_SET(Check RESTRICTED SESSION for TRUE', 'RESTRICTED
SESSION TRUE');
```

With the rule and rule set in place, you are ready to create an ALTER SYSTEM command rule that will check if the `RESTRICTED SESSION` parameter:

```
BEGIN
DBMS_MACADM.CREATE_COMMAND_RULE(
 command         => 'ALTER SYSTEM',
 rule_set_name   => 'Check RESTRICTED SESSION for TRUE',
 object_owner    => '%',
 object_name     => '%',
 enabled         => DBMS_MACUTL.G_YES,
 clause_name     => 'SECURITY',
 parameter_name  => 'RESTRICTED SESSION',
 scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

In this example:

- `rule_set_name` checks whether `RESTRICTED SESSION` is set to `TRUE` or `FALSE`. You must create the rule set and rule in the same location as the command rule: either in the application root or locally in a PDB.
- `object_owner` and `object_name` must always be set to `%` for this kind of `ALTER SESSION` or `ALTER SYSTEM` command rule.
- `enabled` enables you to enable or disable the command rule, or to use simulation mode to log violations to the command rule to a designated log table. The log data describes the error, such as the user name or SQL statement used.
- `clause_name` specifies the `SECURITY` clause of the `ALTER SYSTEM` SQL statement
- `parameter_name` specifies the `RESTRICTED SESSION` parameter from the `SECURITY` clause
- `scope` sets the command rule to be local to the current PDB. The associated rule set and rule must also be local to the current PDB. If you want to create the command rule in the application root, then as a common user, you would set `scope` to `DBMS_MACUTL.G_SCOPE_COMMON` and run the procedure (and its accompanying rule set and rule creation procedures) from the application root.

See Also:

- CREATE_COMMAND_RULE Procedure about the `DBMS_MACADM.CREATE_COMMAND_RULE` procedure
- CREATE_SESSION_EVENT_CMD_RULE Procedure about the `DVS.DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE` procedure
- CREATE_SYSTEM_EVENT_CMD_RULE Procedure for more information about the `DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE` procedure
- DBA_DV_COMMAND_RULE View for information about the `DBA_DV_COMMAND_RULE` data dictionary view
- *Oracle Database SQL Language Reference* for information about the `ALTER SESSION` SQL statement
- *Oracle Database SQL Language Reference* for information about the `ALTER SYSTEM` SQL statement

6.2 Default Command Rules

Oracle Database Vault provides default command rules, based on commonly used SQL statements.

Table 6-1 lists the default Database Vault command rules.

Table 6-1 Default Command Rules

SQL Statement	Rule Set Name
CREATE USER	Can Maintain Accounts/Profiles
ALTER USER	Can Maintain Own Account
DROP USER	Can Maintain Accounts/Profiles
CREATE PROFILE	Can Maintain Accounts/Profiles
ALTER PROFILE	Can Maintain Accounts/Profiles
DROP PROFILE	Can Maintain Accounts/Profiles
ALTER SYSTEM	Allow Fine Grained Control of System Parameters
CHANGE PASSWORD	Can Maintain Own Account[1]

[1] The actual SQL statement that the Can Maintain Own Account rule refers to is PASSWORD.

The following set of command rules helps you to achieve separation of duty for user management:

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

To grant a user the ability to use these commands, you can grant the user the role that the rule set checks. For example, the CREATE USER command rule ensures that a user who tries to run a CREATE USER statement has the DV_ACCTMGR role.

> **Note:**
>
> To find information about the default command rules, query the DBA_DV_COMMAND_RULE data dictionary view.

6.3 SQL Statements That Can Be Protected by Command Rules

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

The SQL statements that you can protect are as follows:

SQL Statements A-A	SQL Statements A-D	SQL Statements C-U
ADMINISTER KEY MANAGEMENT	ANALYZE TABLE	CREATE SYNONYM
ALTER CLUSTER	ASSOCIATE STATISTICS	CREATE TABLE
ALTER DIMENSION	AUDIT	CREATE TABLESPACE

SQL Statements A-A	SQL Statements A-D	SQL Statements C-U
ALTER FLASHBACK ARCHIVE	AUDIT POLICY (for enabling audit unified audit policies)	CREATE TRIGGER
ALTER FUNCTION	CHANGE PASSWORD	CREATE TYPE
ALTER INDEX	COMMENT	CREATE TYPE BODY
ALTER INDEXTYPE	CONNECT	CREATE VIEW
ALTER JAVA	CREATE AUDIT POLICY	DELETE
ALTER LIBRARY	CREATE EDITION	DISASSOCIATE STATISTICS
ALTER OPERATOR	CREATE FLASHBACK ARCHIVE	DROP CLUSTER
ALTER OUTLINE	CREATE USER	DROP CONTEXT
ALTER MATERIALIZED VIEW	CREATE CLUSTER	DROP DATABASE LINK
ALTER MATERIALIZED VIEW LOG	CREATE CONTEXT	DROP EDITION
ALTER PACKAGE	CREATE DATABASE LINK	DROP DIMENSION
ALTER PACKAGE BODY	CREATE DIMENSION	DROP DIRECTORY
ALTER PLUGGABLE DATABASE	CREATE DIRECTORY	DROP FLASHBACK ARCHIVE
ALTER PROCEDURE	CREATE FUNCTION	DROP FUNCTION
ALTER PROFILE	CREATE INDEX	FLASHBACK TABLE
ALTER RESOURCE COST	CREATE INDEXTYPE	EXECUTE
ALTER ROLE	CREATE JAVA	GRANT
ALTER ROLLBACK SEGMENT	CREATE LIBRARY	INSERT
ALTER SEQUENCE	CREATE OPERATOR	NOAUDIT
ALTER SESSION	CREATE OUTLINE	NOAUDIT POLICY (for disabling unified audit policies only)
ALTER SYNONYM	CREATE PACKAGE	PURGE DBA_RECYCLEBIN
ALTER SYSTEM	CREATE PACKAGE BODY	PURGE INDEX
ALTER TABLE	CREATE PLUGGABLE DATABASE	RENAME
ALTER TABLESPACE	CREATE PROCEDURE	PURGE RECYCLEBIN
ALTER TRIGGER	CREATE PROFILE	PURGE TABLE
ALTER TYPE	CREATE ROLE	PURGE TABLESPACE
ALTER TYPE BODY	CREATE ROLLBACK SEGMENT	REVOKE
ALTER USER	CREATE SCHEMA	SELECT
ALTER VIEW	CREATE SEQUENCE	TRUNCATE CLUSTER
ANALYZE CLUSTER	CREATE MATERIALIZED VIEW	TRUNCATE TABLE
ANALYZE INDEX	CREATE MATERIALIZED VIEW LOG	UPDATE

Related Topics

- Command Rules in a Multitenant Environment
  You can create common and local command rules in either the CDB root or the application root.

6.4 Creating a Command Rule

You can create a different types of command rules using different command rule APIs.

Depending on the command rule that you want to create, you can use one of the following command rule APIs to create the command rule: `DBMS_MACADM.CREATE_COMMAND_RULE`, `DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE`, `DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE`. The `DBMS_MACADM.CREATE_COMMAND_RULE` procedure enables you to create complex command rules for `ALTER SYSTEM` and `ALTER SESSION` statements. This topic describes how to create a command rule using the `DBMS_MACADM.CREATE_COMMAND_RULE` procedure.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. If necessary, create a rule set that the command rule will use.

   The `DBA_DV_RULE_SET` data dictionary view lists existing rule sets.
3. Run the `DBMS_MACADM.CREATE_COMMAND_RULE` to create the command rule.

   For example, to create a simple command rule:

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'GRANT',
  rule_set_name   => 'Can Grant VPD Administration',
  object_owner    => 'HR',
  object_name     => 'EMPLOYEES',
  enabled         => DBMS_MACUTL.G_YES,
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

   In this specification:

   - `command` is the SQL statement that you want to protect. The `DBA_DV_COMMAND_RULE` data dictionary view lists the SQL statements that are protected by command rules. If you plan to create a command rule for a unified audit policy object, then ensure that you specify `AUDIT POLICY` or `NOAUDIT POLICY`, not `AUDIT` or `NOAUDIT`, as the command.
     If you want to create a command rule for the `ALTER SYSTEM` or `ALTER SESSION` statements, then you must include a set of special parameters to define the details of these statements: `clause_name`, `parameter_name`, `event_name`, `component_name`, and `action_name`. These parameters, as well as examples of how to use them, are described in the `CREATE_COMMAND_RULE` reference. See Related Topics.
   - `rule_set_name` is the rule set to associate with this command rule. 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. The `DBA_DV_RULE_SET` data dictionary view lists existing rule sets. This parameter is mandatory.
   - `object_owner` is the schema to which this command rule will apply. This attribute is mandatory for all SQL statements that operate on objects within a specific schema. To find the available schema users, query the `DBA_USERS` view. 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`.

The wildcard `%` is not allowed for the command rules for the `SELECT`, `INSERT`, `UPDATE`, `DELETE`, and `EXECUTE` statements. Nor is `%` allowed for `SELECT`, `INSERT`, `UPDATE`, `DELETE`, and `EXECUTE` statements to do a selection of all (`%`) or the `SYS` and `DVSYS` schemas.

- `object_name` is the name of the database object that the command rule affects. Specify `%` to select all database objects, which can include tables, procedures, views, unified audit policies, and so on. This attribute is mandatory if you specified `object_owner`.
- `enabled` controls the status of the command rule. Valid settings are `DBMS_MACUTL.G_YES` `'y'` to enable the command rule (default), `DBMS_MACUTL.G_NO` or `'n'` to disable the command rule, including the capture of violations in the simulation log, or `DBMS_MACUTL.G_SIMULATION` or `'s'` to enable SQL statements to run but capture violations in the simulation log.
- `scope` defines whether the command rule is authorized locally in the current PDB (`DBMS_MACUTL.G_SCOPE_LOCAL`) or in an application root (`DBMS_MACUTL.G_SCOPE_COMMON`). If you create the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example:

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

Related Topics

- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.
- CREATE_COMMAND_RULE Procedure
  The `CREATE_COMMAND_RULE` procedure creates both command and local command rules, which can be added to a rule set.
- SQL Statements That Can Be Protected by Command Rules
  You can protect a large number of SQL statements by using command rules.
- Oracle Database Vault Command Rule APIs
  The `DBMS_MACADM` PL/SQL package provides procedures for configuring command rules. .

6.5 Modifying a Command Rule

You can use the `DBMS_MACADM.UPDATE_COMMAND_RULE`, `DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE`, `DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE`, and `DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE` procedures to modify the definition of a command rule.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Find the command rule and check its definition.

   For example:

   ```
   SELECT COMMAND, ENABLED FROM DBA_DV_COMMAND_RULE ORDER BY COMMAND;
   ```

   The `DBA_DV_COMMAND_RULE` view also shows the definition of the command rule.

3. Run the appropriate procedure to modify the command rule.

   - `DBMS_MACADM.UPDATE_COMMAND_RULE` updates a command rule declaration that was created with the `DBMS_MACADM.CREATE_COMMAND_RULE` procedure. For example:

     ```
     BEGIN
      DBMS_MACADM.UPDATE_COMMAND_RULE(
       command         => 'GRANT',
       rule_set_name   => 'Can Grant VPD Administration',
       object_owner    => 'HR',
       object_name     => 'EMPLOYEES',
       enabled         => DBMS_MACUTL.G_NO,
       scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
     END;
     /
     ```

   - `DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE` updates command rules that were created with `DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE`.
   - `DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE` updates command rules that were created with `DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE`.
   - `DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE` updates command rules that were created with `DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE`.

Related Topics

- Oracle Database Vault Command Rule APIs
  The `DBMS_MACADM` PL/SQL package provides procedures for configuring command rules. .

6.6 Deleting a Command Rule

Before you delete a command rule, you can locate the various references to it by querying the command rule-related Oracle Database Vault views.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_COMMAND_RULE` data dictionary to find the command rule to delete.

   For example:

   ```
   SELECT COMMAND FROM DBA_DV_COMMAND_RULE ORDER BY COMMAND;
   ```

   The `DBA_DV_COMMAND_RULE` view also shows the definition of the command rule.
3. Query the `DBA_DV_COMMAND_RULE` data dictionary to find the definition of the command rule.

   When you drop a command rule, you must omit the `rule_set_name` and `enabled` parameters, and ensure that the rest of the parameters match the settings that were used the last time the command rule was updated.

   For example:

   ```
   SELECT OBJECT_OWNER, OBJECT_NAME, COMMON
   FROM DBA_DV_COMMAND_RULE
   WHERE COMMAND = 'GRANT';
   ```
4. Run the appropriate procedure to delete the command rule.

   - `DBMS_MACADM.DELETE_COMMAND_RULE` deletes a command rule that was created with the `DBMS_MACADM.CREATE_COMMAND_RULE` procedure. For example:

```
BEGIN
 DBMS_MACADM.DELETE_COMMAND_RULE(
  command         => 'GRANT',
  object_owner    => 'HR',
  object_name     => 'EMPLOYEES',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

- `DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE` deletes command rules that were created with `DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE`.
- `DBMS_MACADM.DELETE_SESSION_EVENT_CMD_RULE` deletes command rules that were created with `DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE`.
- `DBMS_MACADM.DELETES_SYSTEM_EVENT_CMD_RULE` deletes command rules that were created with `DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE`.

Related Topics

- Oracle Database Vault Command Rule APIs
  The `DBMS_MACADM` PL/SQL package provides procedures for configuring command rules. .

6.7 How Command Rules Work

Command rules follow a set of steps to check their associated components.

How Realms Work describes what happens when a database account issues a `SELECT`, DDL, or DML statement that affects objects within a realm.

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

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

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

   You can associate rule sets with both command rules and realm authorizations. Oracle Database Vault evaluates the realm authorization rule set first, and then it evaluates the rule sets that apply to the command type being evaluated.
2. For each command rule that applies, Oracle Database Vault evaluates its associated rule set.
3. If the associated rule set of any of the applicable command rules returns false or errors, Oracle Database Vault prevents the command from executing. Otherwise, the command is authorized for further processing. The configuration of the rule set with respect to auditing and event handlers dictates the auditing or custom processing that occurs.

   Command rules override object privileges. That is, even the owner of an object cannot access the object if the object is protected by a command rule. You can disable either a command rule or the rule set of a command. If you disable a command rule, then the command rule does not perform the check it is designed to handle. If you disable a rule set, then the rule set always evaluates to `TRUE`. However, if you want to disable a command rule for a particular command, then you should disable the command rule because the rule set may be associated with other command rules or realm authorizations.

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

In this tutorial, you create a simple local command rule to control whether users can create tables in the `SCOTT` schema.

- Step 1: Create a Table
  First, user `SEBASTIAN` must create a table.
- Step 2: Create a Command Rule
  After the table has been created in the `SEBASTIAN` schema, you can create a command rule.
- Step 3: Test the Command Rule
  Next, you are ready to test the `CREATE TABLE` local command rule.
- Step 4: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

6.8.1 Step 1: Create a Table

First, user `SEBASTIAN` must create a table.

1. Log in to a PDB as the Database Vault Account Manager, who has the `DV_ACCTMGR` role.
2. Create the local user `SEBASTIAN`.

   For example:

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

   Replace *password* with a password that meets the password complexity requirements of the user's profile.
3. Connect as a user who has the `SYSDBA` administrative privilege and then grant the following additional privilege to `SEBASTIAN`.

   ```
   GRANT CREATE TABLE TO SEBASTIAN;
   ```

4. As user `SEBASTIAN`, create a table.

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

5. Now drop the table.

   ```
   DROP TABLE t1;
   ```

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

Related Topics

- *Oracle Database Security Guide*

6.8.2 Step 2: Create a Command Rule

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

1. Connect to a PDB as a user who has been granted the DV_OWNER role.
2. Create a CREATE TABLE command rule with user SEBASTIAN as the owner.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'CREATE TABLE',
  rule_set_name   => 'Disabled',
  object_owner    => 'SEBASTIAN',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

This command rule will prevent user SEBASTIAN from creating tables in their schema, even though he is the schema owner. The object_name will apply the command rule to all objects in the SEBASTIAN schema.

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

6.8.3 Step 3: Test the Command Rule

Next, you are ready to test the CREATE TABLE local command rule.

1. In SQL*Plus, ensure that you are logged in to the PDB as user SEBASTIAN.
2. Try to create a table.

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

The following output should appear:

```
ORA-47400: Command Rule violation for create table on SEBASTIAN.T1
```

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

3. Now enable user SEBASTIAN to create tables again.
   a. Connect to the PDB as the user who created the command rule.
   b. Update the CREATE TABLE command rule to now enable table creations.

```
BEGIN
 DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'CREATE TABLE',
  rule_set_name   => 'Enabled',
  object_owner    => 'SEBASTIAN',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

4. Connect as user SEBASTIAN, and then try creating the table again.

```
CREATE TABLE t1 (num NUMBER);

Table created.
```

5. User SEBASTIAN does not really need this table, so drop the table.

```
DROP TABLE t1;
```

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

6.8.4 Step 4: Remove the Components for This Tutorial

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

1. Connect to the PDB as the user who created the CREATE TABLE command rule.
2. Drop the CREATE TABLE command rule.

   Remember that the command, object_owner, and object_name arguments must match exactly the arguments that were used the last time the command rule was updated. You can check a command rule's definition by querying the DBA_DV_COMMAND_RULE data dictionary view.

```
BEGIN
 DBMS_MACADM.DELETE_COMMAND_RULE(
  command         => 'CREATE TABLE',
  object_owner    => 'SEBASTIAN',
  object_name     => '%');
END;
/
```

3. As a user who has the DV_ACCTMGR role, drop user SEBASTIAN.

```
DROP USER SEBASTIAN;
```

6.9 Guidelines for Designing Command Rules

Oracle provides guidelines for designing command rules.

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

  For example, if you want to prevent SELECT statements from occurring on specific schema objects, then design multiple command rules to stop the SELECT statements on those specific schema objects, rather than creating a general command rule to prevent SELECT statements in the schema level.

- When designing rules for the CONNECT event, be careful to include logic that does not inadvertently lock out any required user connections. If any account has been locked out accidentally, ask a user who has been granted the DV_ADMIN or DV_OWNER role to log in and correct the rule that is causing the lock-out problem. The CONNECT command rule does not apply to users with the DV_OWNER and DV_ADMIN roles. This prevents improperly configured CONNECT command rules from causing a complete lock-out.

  If the account has been locked out, you can disable Oracle Database Vault, correct the rule that is causing the lock-out problem, and then reenable Oracle Database Vault. Even

when Oracle Database Vault is disabled, you still can use Database Vault Administrator and the Database Vault PL/SQL packages.

- If you must temporarily relax an enabled command rule for an administrative task, then consider switching the command rule to simulation mode. Note that this will not capture activity that meets the rule set criteria, only activity that would have violated it.
- When designing command rules, be careful to consider automated processes such as backup where these procedures may be inadvertently disabled. You can account for these tasks by creating rules that allow the command when a series of Oracle Database Vault factors is known to be true (for example, the program being used), and the account being used or the computer or network on which the client program is running.
- You can test the development phase of a command rule by using simulation mode, which enables the command rule but writes detailed information about it to a log file.

Related Topics

- Using Simulation Mode for Logging Realm and Command Rule Activities
  Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

6.10 How Command Rules Affect Performance

The performance of a command rule depends on the complexity of the rules in the rule set associated with the command rule.

For example, suppose a rule set invokes a PL/SQL function that takes 5 seconds to run. In this case, a command rule that uses that rule set would take 5 seconds to grant access for the command statement to run.

You can check the system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Cloud Control, which is installed by default with Oracle Database), Automatic Workload Repository (AWR), and `TKPROF`.

Related Topics

- *Oracle Database Performance Tuning Guide*
- *Oracle Database SQL Tuning Guide*

6.11 Command Rule Related Reports and Data Dictionary View

Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing command rules.

Table 6-2 lists the Oracle Database Vault report.

Table 6-2 Reports Related to Command Rules

Report	Description
Command Rule Audit Report	Lists audit records generated by command rule processing operations
Command Rule Configuration Issues Report	Tracks rule violations, in addition to other configuration issues the command rule may have
Object privilege reports	List object privileges that the command rule affects
Sensitive object reports	List objects that the command rule affects
Rule Set Configuration Issues Report	Lists rules sets that have no rules defined or enabled, which may affect the command rules that use them

You can use the `DBA_DV_COMMAND_RULE` data dictionary view to find the SQL statements that are protected by command rules.

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

7

Configuring Factors

Factors allow you to create and use complex attributes through PL/SQL to make Oracle Database Vault authorization decisions.

- What Are Factors?
  A factor is a named variable or attribute, such as a database IP address, that Oracle Database Vault can recognize.
- Default Factors
  Oracle Database Vault provides a set of default factors.
- Creating a Factor
  In general, to create a factor, you first create the factor itself, and then you edit the factor to include its identity.
- Adding an Identity to a Factor
  After you create a new factor, you optionally can add an identity to it.
- Modifying a Factor
  You can use the `DBMS_MACADM.UPDATE_FACTOR` procedure to modify the definition of a factor.
- Deleting a Factor
  Before you delete a factor, you must remove references to the factor.
- How Factors Work
  Oracle Database Vault processes factors when a session is established.
- Tutorial: Preventing Ad Hoc Tool Access to the Database
  This tutorial demonstrates how to use factors to prevent ad hoc tools (such as SQL*Plus) from accessing the database.
- Guidelines for Designing Factors
  Oracle provides guidelines for designing factors.
- How Factors Affect Performance
  The complexity of factors affects the performance of your Oracle database instance.
- Factor Related Reports and Data Dictionary Views
  Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing factors and their identities.

7.1 What Are Factors?

A factor is a named variable or attribute, such as a database IP address, that Oracle Database Vault can recognize.

You can use factors for activities such as authorizing database accounts to connect to the database or creating filtering logic to restrict the visibility and manageability of data.

Oracle Database Vault provides a selection of factors that lets you set controls on such components as the domain for your site, IP addresses, databases, and so on. You also can create custom factors, using your own PL/SQL retrieval methods. However, for the vast majority of cases, you can use the `SYS_CONTEXT` PL/SQL function to create rules on the most

commonly used factors that are readily available in the database. Such factors as `Session_User`, `Proxy_User`, `Network_Protocol`, and `Module` are available through the `SYS_CONTEXT` function.

Factors have powerful capabilities that are used in conjunction with Oracle Label Security and for other database attributes that are not already available through context parameters. Commonly available factors are listed in this section, but Oracle recommends that you use the `SYS_CONTEXT` function in the rule definitions for these factors. Only create and use factors that are not already available through `SYS_CONTEXT`.

Note the following:

- You can use factors in combination with rules in rule sets. The `DVF` factor functions are factor-specific functions that you can use in rule expressions.
- Factors have values (identities) and are further categorized by their factor types.
- You also can integrate factors with Oracle Label Security labels.
- You can run reports on the factors that you create in Oracle Database Vault.
- You only can create factors in a PDB, not in the CDB root or the application root.

Related Topics

- Creating a Rule to Add to a Rule Set
  A rule defines the behavior that you want to control; a rule set is a named collection of rules.
- *Oracle Database SQL Language Reference*
- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the `DVF` schema functions when you use the `DBMS_MACADM` PL/SQL package to manage the various factors.
- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

7.2 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.

You can create custom factors by using your own PL/SQL retrieval methods. A useful PL/SQL function you can use (which is used for many of the default factors) is the `SYS_CONTEXT` SQL function, which retrieves data about the user session. For example, you can use the `CLIENT_PROGRAM_NAME` attribute of `SYS_CONTEXT` to find the name of the program used for the database session. After you create the custom factor, you can query its values similar to the functions used to query the default factors.

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

The default factors are as follows:

- Authentication_Method is the method of authentication. In the list that follows, the type of user is followed by the method returned:
  - Password-authenticated enterprise user, local database user, user with the SYSDBA or SYSOPER administrative privilege using the password file; proxy with user name using password: PASSWORD
  - Kerberos-authenticated enterprise user or external user (with no administrative privileges): KERBEROS
  - Kerberos-authenticated enterprise user (with administrative privileges): KERBEROS_GLOBAL
  - Kerberos-authenticated external user (with administrative privileges): KERBEROS_EXTERNAL
  - Transport Layer Security (TLS)-authenticated enterprise or external user (with no administrative privileges): SSL (Transport Layer Security replaces Secure Sockets Layer, but SSL-related settings will work with Transport Layer Security.)
  - Transport Layer Security-authenticated enterprise user (with administrative privileges): SSL_GLOBAL
  - Transport Layer Security-authenticated external user (with administrative privileges): SSL_EXTERNAL
  - Radius-authenticated external user: RADIUS
  - OS-authenticated external user, or user with the SYSDBA or SYSOPER administrative privilege: OS
  - Proxy with certificate, DN, or username without using password: NONE
  - Background process (job queue secondary process): JOB
  - Parallel Query Slave (secondary) process: PQ_SLAVE

  For non-administrative connections, you can use the Identification_Type factor to distinguish between external and enterprise users when the authentication method is PASSWORD, KERBEROS, or SSL. For administrative connections, the Authentication_Method factor is sufficient for the PASSWORD, SSL_EXTERNAL, and SSL_GLOBAL authentication methods.
- Client Identifier is an identifier that is set by the application through the DBMS_SESSION.SET_IDENTIFIER procedure, the Oracle Call Interface (OCI) attribute OCI_ATTR_CLIENT_IDENTIFIER, or Oracle Dynamic Monitoring Service (DMS). Various Oracle Database components use this attribute to identify lightweight application users who authenticate as the same database user.
- Client_IP is the IP address of the machine from which the client is connected.
- Database_Domain is the domain of the database as specified in the DB_DOMAIN initialization parameter.
- Database_Hostname is the host name of the computer on which the instance is running.
- Database_Instance is the instance identification number of the current instance.
- Database_IP is the IP address of the computer on which the instance is running.
- Database_Name is the name of the database as specified in the DB_NAME initialization parameter.
- DBlink_Info is the source of a database link session. The string has this form:

SOURCE_GLOBAL_NAME=*dblink_src_global_name*, DBLINK_NAME=*dblink_name*,SOURCE_AUDIT_SESSIONID=*dblink_src_audit_sessionid*

In this specification:

- *dblink_src_global_name* is the unique global name of the source database
- *dblink_name* is the name of the database link on the source database
- *dblink_src_audit_sessionid* source database that initiated source database that initiated the connection to the remote database using *dblink_name*

- `Domain` is a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level. You can identify a domain using factors such as host name, IP address, and database instance names of the Database Vault nodes in a secure access path to the database. Each domain can be uniquely determined using a combination of the factor identifiers that identify the domain. You can use these identifying factors and possibly additional factors to define the Maximum Security Label within the domain. This restricts data access and commands, depending on the physical factors about the Database Vault session. Example domains of interest may be Corporate Sensitive, Internal Public, Partners, and Customers.
- `Enterprise_Identity` is the enterprise-wide identity for the user:
  - For enterprise users: the Oracle Internet Directory-distinguished name (DN).
  - For external users: the external identity (Kerberos principal name, Radius and DCE schema names, operating system user name, certificate DN).
  - For local users and `SYSDBA` and `SYSOPER` logins: NULL.

  The value of the attribute differs by proxy method:
  - For a proxy with DN: the Oracle Internet Directory DN of the client.
  - For a proxy with certificate: the certificate DN of the client for external users; the Oracle Internet Directory DN for global users.
  - For a proxy with user names: the Oracle Internet Directory DN if the client is an enterprise user; NULL if the client is a local database user.
- `Identification_Type` is the way the user schema was created in the database. Specifically, it reflects the `IDENTIFIED` clause in the `CREATE USER` and `ALTER USER` syntax. In the list that follows, the syntax used during schema creation is followed by the identification type returned:
  - `IDENTIFIED BY` *password*: `LOCAL`
  - `IDENTIFIED EXTERNALLY`: `EXTERNAL`
  - `IDENTIFIED GLOBALLY`: `GLOBAL SHARED`
  - `IDENTIFIED GLOBALLY AS DN`: `GLOBAL PRIVATE`
  - `GLOBAL EXCLUSIVE` for exclusive global user mapping
  - `GLOBAL SHARED` for shared user mapping
  - `NONE` when the schema is created with no authentication
- `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`

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

Related Topics

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

7.3 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.

This procedure explains how to create the factor only, not how to configure an identity for it.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. If necessary, create a rule set that the factor will use.

   The `DBA_DV_RULE_SET` data dictionary view lists existing rule sets.
3. Run the `DBMS_MACADM.CREATE_FACTOR` procedure to create the factor.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => 'Factor to restrict DBA access',
  rule_set_name     => 'Limit_DBA_Access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
```

```
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_METHOD,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_SESSION,
  audit_options     => DBMS_MACUTL.G_AUDIT_OFF,
  fail_options      => DBMS_MACUTL.G_FAIL_SILENTLY);
END;
/
```

In this specification:

- `factor_name` can be up to 128 characters in mixed-case, without spaces. The the `DBA_DV_FACTOR` data dictionary view lists existing factors. This parameter is mandatory.
- `factor_type_name` can be up to 128 characters in mixed-case, without spaces. The `DBA_DV_FACTOR_TYPE` data dictionary view lists existing factor types. This parameter is mandatory. The
  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. If you want to find factors that are associated with a particular factor type, query the `DBA_DV_FACTOR` view. For example:

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

- `description` can have up to 1024 characters in mixed-case. This parameter is mandatory.
- `rule_set_name` is the rule set name if you want to use a rule set to control when and how a factor identity is set. The `DBA_DV_RULE_SET` data dictionary view lists rules sets. This parameter is mandatory.
  This setting is particularly useful for situations where database applications, such as a Web application using a JDBC connection pool, must dynamically set a factor identity for the current database session. For example, a Web application may want to assign the geographic location for a database account logging in to the Web application. To do so, the Web application can use the JDBC Callable Statement, or Oracle Data Provider for .NET (ODP.NET) to execute the PL/SQL function `SET_FACTOR`, for example:

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

  Then you can create an assignment rule for the `GEO_STATE` factor to allow or disallow the setting of the `GEO_STATE` factor based on other factors or rule expressions.
- `get_expr` is a valid PL/SQL expression that retrieves the identity of a factor. It can use up to 255 characters in mixed-case. The following retrieval method sets a value of the `DB_NAME` factor by retrieving the database name (`DB_NAME`) from the `USERENV` namespace in a user's session:

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

- `validate_expr` is a valid PL/SQL expression that returns a Boolean value (`TRUE` or `FALSE`) to validate the identity of the factor being retrieved (with the

GET_FACTOR function) or the value to be assigned to a factor (with the SET_FACTOR function). It can have up to 255 characters and be in mixed case. This parameter is mandatory.
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 feature provides an additional level of assurance that the factor is properly retrieved and set. You can include any package function or standalone function in the expression. Ensure that the expression is a fully qualified function, such as *schema.function_name*. Do not include complete SQL statements. If you are using application packages or functions, then you must provide DVSYS with the EXECUTE privilege on the object.

The PL/SQL expression can use either of these 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.

- identify_by can be one of the following options for determining the identity of a factor, based on the expression set for the get_expr parameter:
  - DBMS_MACUTL.G_IDENTIFY_BY_CONSTANT: By constant
  - DBMS_MACUTL.G_IDENTIFY_BY_METHOD: By method. For example, suppose the expression returns the date and time of the PDB using the time zone of the PDB: to_char(sysdate,'yyyy-mm-dd'). On December 15, 2022, the following value would be returned: 2022-12-15
  - DBMS_MACUTL.G_IDENTIFY_BY_FACTOR: By factor. This setting 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 the get_expr parameter for this option.)
  - DBMS_MACUTL.G_IDENTIFY_BY_CONTEXT: By context
- labeled_by controls how the factor identity retrieves an Oracle Label Security (OLS) label. This parameter is mandatory if you are using the Oracle Label Security integration.
  - DBMS_MACUTL.G_LABELED_BY_SELF labels the identities for the factor directly from the labels associated with an Oracle Label Security policy (default)
  - DBMS_MACUTL.G_LABELED_BY_FACTORS derives the factor identity label from the labels of its child factor identities.
- eval_options evaluate the factor when the user logs on. This parameter is mandatory.
  - DBMS_MACUTL.G_EVAL_ON_SESSION evaluates the factor when the database session is created (default). Be aware that this setting may affect performance of the factor.
  - DBMS_MACUTL.G_EVAL_ON_ACCESS evaluates the factor each time the factor is accessed.
  - DBMS_MACUTL.G_EVAL_ON_STARTUP evaluates the factor on start-up.

- `audit_options` applies only to traditional auditing, not unified auditing environments. Starting with Oracle Database release 23c, traditional auditing is desupported. Oracle recommends that you create unified audit policies instead of using `audit_options`.Valid options for `audit_options` are as follows:
  - `DBMS_MACUTL.G_AUDIT_OFF` disables auditing
  - `DBMS_MACUTL.G_AUDIT_ALWAYS` always audits.
  - `DBMS_MACUTL.G_AUDIT_ON_GET_ERROR` audits if `get_expr` returns an error.
  - `DBMS_MACUTL.G_AUDIT_ON_GET_NULL` audits if `get_expr` is null.
  - `DBMS_MACUTL.G_AUDIT_ON_VALIDATE_ERROR` audits if the validation procedure returns an error.
  - `DBMS_MACUTL.G_AUDIT_ON_VALIDATE_FALSE` audits if the validation procedure is false.
  - `DBMS_MACUTL.G_AUDIT_ON_TRUST_LEVEL_NULL` audits if there is no trust level set.
  - `DBMS_MACUTL.G_AUDIT_ON_TRUST_LEVEL_NEG` audits if the trust level is negative.
- `fail_options` sets options for reporting factor errors.
  - `DBMS_MACUTL.G_FAIL_WITH_MESSAGE` shows an error message (default).
  - `DBMS_MACUTL.G_FAIL_SILENTLY` does not show an error message.

At this stage, the factor is complete and can be used. For more detailed and customized processing, you can configure an identity for the factor.

Related Topics

- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.
- Adding an Identity to a Factor
  After you create a new factor, you optionally can add an identity to it.
- About Identity Mapping
  While you are creating a factory identity, you can map it.
- How Factors Affect Performance
  The complexity of factors affects the performance of your Oracle database instance.
- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

7.4 Adding an Identity to a Factor

After you create a new factor, you optionally can add an identity to it.

- About Factor Identities
  An identity is the actual value of a factor, such an IP_Address factor identity being 192.0.2.4.
- How Factor Identities Work
  A factor identity is the actual value of a factor (for example, the IP address for a factor that uses the IP_Address type).
- About Trust Levels
  Trust levels enable you to assign a numeric value to indicate the measure of trust allowed.
- About Label Identities
  You can assign You Oracle Label Security (OLS) labels to factor identities.
- Creating and Configuring a Factor Identity
  You can create and configure a factor identity for an existing factor.
- Using Identity Mapping to Configure an Identity to Use Other Factors
  You can use identity mapping to use a group of factors to manage identity values.
- Modifying a Factor Identity
  You can use the `DBMS_MACADM.UPDATE_IDENTITY` procedure to modify a factor identity.
- Deleting a Factor Identity
  Before delete a factor identity, you must remove references to it.

7.4.1 About Factor Identities

An identity is the actual value of a factor, such an IP_Address factor identity being 192.0.2.4.

A factor identity for a given database session is assigned at run time using the `get_expr` parameter (to retrieve the identity of a factor) and the `identify_by` parameter (to determine the identify of the factor) in the `DBMS_MACADM.CREATE_FACTOR` procedure. 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

7.4.2 How Factor Identities Work

A factor identity is the actual value of a factor (for example, the IP address for a factor that uses the IP_Address type).

A factor can have several identities depending on its retrieval method or its identity mapping logic. For example, a factor such as `Database_Hostname` could have multiple identities in an Oracle Real Application Clusters environment; a factor such as `Client_IP` can have multiple identities in any database environment. The retrieval method for these types of factors may return different values because the retrieval method is based on the database session. Several reports allow you to track the factor identity configuration.

You can configure the assignment of a factor in the following ways:

- Assign the factor at the time a database session is established.
- Configure individual requests to retrieve the identity of the factor.

With the Oracle Label Security integration, you can label identities with an Oracle Label Security label. You can also assign an identity *trust levels*, which are numbers that indicate the magnitude of trust relative to other identities for the same factor. In general, the higher the trust level number is set, the greater the trust. Negative trust levels are not trusted.

Within a database session, a factor assigned identity is available to Oracle Database Vault and any application with a publicly accessible PL/SQL function that exists in the `DVF` schema (which contains functions that retrieve factor values) as follows:

```
dvf.f$factor_name
```

This allows the identifier for a factor to be accessed globally from within the Oracle database (using PL/SQL, SQL, Oracle Virtual Private Database, triggers, and so on). For example, in SQL*Plus:

```
CONNECT sec_admin_owen@pdb_name
Enter password: password

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

Output similar to the following appears:

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

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

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

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

The following output appears:

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

Related Topics

- Adding an Identity to a Factor
  After you create a new factor, you optionally can add an identity to it.
- Factor Related Reports and Data Dictionary Views
  Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing factors and their identities.

7.4.3 About Trust Levels

Trust levels enable you to assign a numeric value to indicate the measure of trust allowed.

A trust value of 1 signifies some trust. A higher value indicates a higher level of trust. A negative value or zero indicates distrust. When the factor identity returned from a factor retrieval method is not defined in the identity, Oracle Database Vault automatically assigns the identity a negative trust level.

To determine the trust level of a factor identity at run time, you can use the GET_TRUST_LEVEL and GET_TRUST_LEVEL_FOR_IDENTITY functions in the DVSYS schema.

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

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

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

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

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

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

Output similar to the following appears:

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

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

Related Topics

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

7.4.4 About Label Identities

You can assign You Oracle Label Security (OLS) labels to factor identities.

In brief, a label acts as an identifier for a database table row to assign privileges to the row. In the DBMS_MACADM.CREATE_FACTOR or DBMS_MACADM.UPDATE_FACTOR procedure, the labeled_by parameter setting determines whether a factor is labeled DBMS_MACUTL.G_LABELED_BY_SELF or DBMS_MACUTL.G_LABELED_BY_FACTORS. If you set labeled_by to DBMS_MACUTL.G_LABELED_BY_SELF, then you can associate OLS labels with the factor identities. If you set labeled_by to DBMS_MACUTL.G_LABELED_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.

Related Topics

- *Oracle Label Security Administrator's Guide*

7.4.5 Creating and Configuring a Factor Identity

You can create and configure a factor identity for an existing factor.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Run the `DBMS_MACADM.CREATE_IDENTITY` procedure.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_IDENTITY(
  factor_name  => 'Sector2_ClientID',
  value        => 'intranet',
  trust_level  => 5);
END;
/
```

   In this specification:

   - `factor_name` is the name of the existing factor. The `DBA_DV_FACTOR` data dictionary view lists factors.
   - `value` is the 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` 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.
     - `10` is very trusted.
     - `5` is trusted.
     - `1` is somewhat trusted.
     - `-1` is untrusted.
     - `NULL` is for a trust level that is not defined (default)

After you create a factor identity, you can use it in an identity map with two existing factors.

Related Topics

- 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.
- Mapping an Identity to a Factor
  You can map an identity to a factor by creating a parent-child relationship with two factors.
- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

7.4.6 Using Identity Mapping to Configure an Identity to Use Other Factors

You can use identity mapping to use a group of factors to manage identity values.

- About Identity Mapping
  While you are creating a factory identity, you can map it.
- Mapping an Identity to a Factor
  You can map an identity to a factor by creating a parent-child relationship with two factors.
- Deleting an Identity Map
  To remove the parent-child relationship between two factors, you must delete the identity map.

7.4.6.1 About Identity Mapping

While you are creating a factory identity, you can map it.

Identity mapping is the process of identifying a factor by using other (child) factors. This is a way to transform combinations of factors into logical identities for a factor or to transform continuous identity values (for example, temperature) or large discrete identity values (for example, IP address ranges) into logical sets. To check configuration issues in the mapping for an identity, you can run the Identity Configuration Issues report.

You can map different identities of a parent factor to different identities of the contributing factor. For example, an `INTRANET` identity maps to an IP address range of 192.0.2.1 to 192.0.2.24. A `REMOTE` identity can map to an IP address range that excludes the address range 192.0.2.1 to 192.0.2.24.

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

If you need to change the identity mapping, you must delete and then recreate the identity map.

7.4.6.2 Mapping an Identity to a Factor

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

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Ensure that you have created the factors and factor identities that you that you plan to use for the mapping.

   The `DBA_DV_FACTOR` data dictionary view lists the existing factors. The `DBA_DV_IDENTITY` data dictionary view lists the existing factor identities.
3. Run the `DBMS_MACADM.CREATE_IDENTITY_MAP` procedure to create the identity map.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_IDENTITY_MAP(
  identity_factor_name  => 'Sector2_ClientID',
  identity_factor_value => 'intranet',
  parent_factor_name    => 'HQ_ClientID',
  child_factor_name     => 'Div1_ClientID',
```

```
  operation             => '<',
  operand1              => '192.0.2.50',
  operand2              => '192.0.2.100');
END;
/
```

In this specification:

- `identity_factor_name` is the factor to be used for the identity map.
- `identity_factor_value` is value the factor assumes if the identity map evaluates to `TRUE`.
- `parent_factor_name` is the parent factor link to which the map is related. The `DBA_DV_IDENTITY_MAP` data dictionary view lists existing parent-child factor mappings.
- `child_factor_name` is the child factor link to which the map is related.
- `operation` is a relational operator for the identity map (for example, `<`, `>`, `=`, `between`, and so on).
- `operand1` is the left operand for the relational operator and refers to the low value you enter.
- `operand2` is the right operand for the relational operator and refers to the high value you enter.

For example, consider a scenario where the child factor is set to `Client_IP`, `operation` is set to `between`, `operand1` is set to `192.0.2.1`, and `operand2` 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`).

4. Repeat this process to add more contributing factors for a parent factor identity.

   For example, you can configure the `Network` factor to resolve to a value `ACCOUNTING-SENSITIVE`, when the `Program` factor resolves to `Oracle General Ledger` and the `Client_IP` is in between `192.0.2.1` and `192.0.2.24`. So, if an authorized accounting financial application program, running on a client with IP address `192.0.2.12` accesses the database, then the `Network` factor is resolved to `ACCOUNTING-SENSITIVE`. A database session with the `ACCOUNTING-SENSITIVE Network` value would have more access privileges than one with the `INTRANET Network` value.

Related Topics

- 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.
- Creating and Configuring a Factor Identity
  You can create and configure a factor identity for an existing factor.
- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

7.4.6.3 Deleting an Identity Map

To remove the parent-child relationship between two factors, you must delete the identity map.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_FACTOR_LINK` data dictionary view to find the factors that have been used in parent-child mappings.

   For example:

```
SELECT PARENT_FACTOR_NAME, CHILD_FACTOR_NAME FROM DBA_DV_FACTOR_LINK;

PARENT_FACTOR_NAME             CHILD_FACTOR_NAME
------------------------------ ------------------------------
Domain                         Database_Instance
Domain                         Database_IP
Domain                         Database_Hostname
```

3. Query the `DBA_DV_IDENTITY_MAP` data dictionary view to find the definition of the mapping that you want to remove.
4. Based on the definition of the mapping, run the `DBMS_MACADM.DELETE_IDENTITY_MAP` procedure.

   For example:

```
BEGIN
 DBMS_MACADM.DELETE_IDENTITY_MAP(
  identity_factor_name  => 'intranet-factor',
  identity_factor_value => 'intranet',
  parent_factor_name    => 'Domain',
  child_factor_name     => 'Database_IP',
  operation             => 'between',
  operand1              => '192.0.2.22',
  operand2              => '192.0.2.99');
END;
/
```

Related Topics

- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

7.4.7 Modifying a Factor Identity

You can use the `DBMS_MACADM.UPDATE_IDENTITY` procedure to modify a factor identity.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Find the factor identity and check its definition.

   For example:

```
SELECT * FROM DBA_DV_IDENTITY ORDER BY FACTOR_NAME;
```

3. Run the `DBMS_MACADM.UPDATE_IDENTITY` statement.

   For example:

```
BEGIN
 DBMS_MACADM.UPDATE_IDENTITY(
  factor_name  => 'Sector2_ClientID',
  value        => 'intranet',
  trust_level  => 7);
END;
/
```

Related Topics

- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

7.4.8 Deleting a Factor Identity

Before delete a factor identity, you must remove references to it.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_IDENTITY` data dictionary view to find the factor identity to remove.

   For example:

   ```
   SELECT * FROM DBA_DV_IDENTITY ORDER BY FACTOR_NAME;
   ```

3. Run the `DBMS_MACADM.DELETE_IDENTITY` procedure.

   You must include the `factor_name` and `value` parameters. For example:

   ```
   BEGIN
    DBMS_MACADM.DELETE_IDENTITY(
     factor_name => 'Sector2_ClientID',
     value       => 'intranet');
   END;
   /
   ```

Related Topics

- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

7.5 Modifying a Factor

You can use the `DBMS_MACADM.UPDATE_FACTOR` procedure to modify the definition of a factor.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Find the factor and check its definition.

   For example:

   ```
   SELECT * FROM DBA_DV_FACTOR ORDER BY NAME;
   ```

3. Run the `DBMS_MACADM.UPDATE_FACTOR` statement.

   For example:

```
BEGIN
 DBMS_MACADM.UPDATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => 'Factor to restrict DBA access in Sector2_DB',
  rule_set_name     => 'Limit_DBA_Access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_METHOD,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_ACCESS,
  audit_options     => DBMS_MACUTL.G_AUDIT_OFF,
  fail_options      => DBMS_MACUTL.G_FAIL_WITH_MESSAGE);
END;
/
```

After you update a factor, if it has traditional auditing set (`audit_options`), then this auditing disabled because traditional auditing is desupported starting in release 23c. Only the `DBMS_MACUTL.G_AUDIT_OFF` setting is available for the `audit_options` parameter. To audit a factor, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a factor.

Related Topics

- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

7.6 Deleting a Factor

Before you delete a factor, you must remove references to the factor.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_FACTOR` data dictionary to find the name of the factor to delete.

   ```
   SELECT NAME FROM DBA_DV_FACTOR ORDER BY NAME;
   ```

3. Query the `DBA_DV_FACTOR_LINK` data dictionary to find if the factor is a parent or a child factor.

   For example, assuming the factor is named `Sector2_DB`:

   ```
   SELECT PARENT_FACTOR_NAME, CHILD_FACTOR_NAME
   FROM DBA_DV_FACTOR_LINK
   WHERE PARENT_FACTOR_NAME = 'Sector2_DB'
   OR CHILD_FACTOR_NAME = 'Sector2_DB';
   ```

4. If the factor is a parent or child factor, then delete the factor link.

   For example:

   ```
   BEGIN
    DBMS_MACADM.DELETE_FACTOR_LINK(
     parent_factor_name => 'Sector2_DB',
     child_factor_name  => 'Div1_ClientID');
   END;
   /
   ```

5. Query the `DBA_DV_IDENTITY_MAP` data dictionary view to find the definition for any identity maps that may use the factor.

   For example:

   ```
   SELECT * FROM DBA_DV_IDENTITY_MAP
   WHERE FACTOR_NAME = 'Sector2_DB';
   ```

6. Run the `DBMS_MACADM.DELETE_IDENTITY_MAP` to delete the identity map.

   For example:

   ```
   BEGIN
    DBMS_MACADM.DELETE_IDENTITY_MAP(
     identity_factor_name  => 'Sector2_DB',
     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;
   /
   ```

7. Query the `DBA_DV_IDENTITY` data dictionary view to find if the factor has a reference to any factor identities.

   Query for the factor name and the value. For example:

   ```
   SELECT VALUE FROM DBA_DV_IDENTITY
   WHERE FACTOR_NAME = 'Sector2_DB'
   ```

8. Run the `DBMS_MACADM.DELETE_IDENTITY` procedure to remove the factor reference.

   You must include both the `factor_name` and `value` parameters. For example:

   ```
   BEGIN
    DBMS_MACADM.DELETE_IDENTITY(
     factor_name => 'Sector2_DB',
     value       => 'intranet');
   END;
   /
   ```

9. Run the `DBMS_MACADM.DELETE_FACTOR` to delete the factor.

   For example:

   ```
   EXEC DBMS_MACADM.DELETE_FACTOR('Sector2_DB');
   ```

Related Topics

- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

7.7 How Factors Work

Oracle Database Vault processes factors when a session is established.

- How Factors Are Processed When a Session Is Established
  Oracle Database Vault evaluates the factors based on when a session begins.

- How Retrieval Methods Work
  The Retrieval Method identifies factors where the factor identification is by method or constant.
- How Factors Are Retrieved
  You can retrieve a factor in a database session at any time by using the `DVF` factor function or the `GET_FACTOR` function.
- How Factors Are Set
  You can assign a factor identity at any time during a database session, but only if the factor assignment rule set evaluates to true.
- How Factor Auditing Works
  Whether you have unified auditing enabled affects how auditing is handled for factors.

7.7.1 How Factors Are Processed When a Session Is Established

Oracle Database Vault evaluates the factors based on when a session begins.

When a database session is established, the following actions occur:

1. At the start of each database session, Oracle Database Vault begins to evaluate all default and user-created factors in the database instance.

   This evaluation occurs after the normal database authentication of the session and the initialization of the Oracle Label Security session information, if applicable.
2. In the factor evaluation stage, the factor initialization process executes the retrieval method for all factors that are identified by methods or constants, to resolve the factor identity for the session.

   The factor error options setting has no effect on the factor initialization process.
3. If a factor has a validation method defined, Oracle Database Vault validates the identity (value) of the factor by executing this validation method. If the validation method fails or returns false, the identity of the factor is undefined (`NULL`).
4. If a factor has any identities defined for it, Oracle Database Vault resolves the trust level of the factor based on the identities defined. If an identity of the factor is defined in this list of defined identities, then Oracle Database Vault assigns the trust level as configured; otherwise it sets it to `-1`. If there are no identities defined for the factor, the trust level is undefined (`NULL`).
5. Depending on the outcome of this factor evaluation, factor validation, and trust level resolution, Database Vault audits the details of the evaluation as dictated by the factor audit configuration.
6. When the evaluation of all factors that are identified by method or constant completes, Oracle Database Vault resolves the factors that are identified by other factors by using the identity maps that are defined for the factor configured identities.

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

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

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

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

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

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

7.7.2 How Retrieval Methods Work

The Retrieval Method identifies factors where the factor identification is by method or constant.

If the factor identification is by factors, Oracle Database Vault identifies it by its identity mappings. You can create your own PL/SQL retrieval methods, or use the functions supplied with Oracle Database Vault. Oracle Database Vault provides factor-specific and general utility functions that you can use to build the retrieval method.

See also the default factors provided with Oracle Database Vault for examples of retrieval methods.

The get_expr parameter is mandatory if you have selected the following DBMS_MACADM.CREATE_FACTOR or DBMS_MACADM.CREATE_UPDATE settings for the identify_by parameter:

- DBMS_MACUTL.G_IDENTIFY_BY_METHOD: Enter a method for the get_expr parameter.
- DBMS_MACUTL.G_IDENTIFY_BY_CONSTANT: Enter a constant for the get_expr parameter.

The value returned as the factor identity must be a VARCHAR2 string or otherwise convertible to one.

You can include any package function or standalone function in the expression. Ensure that the expression is a fully qualified function, such as *schema.function_name*. Do not include complete SQL statements. If you are using application packages or functions, you must provide DVSYS with the EXECUTE privilege on the object.

Write the function signature using the following format:

```
FUNCTION GET_FACTOR RETURN VARCHAR2
```

Related Topics

- Default Factors
  Oracle Database Vault provides a set of default factors.
- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the DVF schema functions when you use the DBMS_MACADM PL/SQL package to manage the various factors.

- DBMS_MACADM Factor Procedures and Functions
  The DBMS_MACADM PL/SQL package provides procedures and functions to configure factors.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the DBMS_MACUTL PL/SQL package.

7.7.3 How Factors Are Retrieved

You can retrieve a factor in a database session at any time by using the DVF factor function or the GET_FACTOR function.

To find a listing of available factors, query the DBA_DV_FACTOR data dictionary view, described in .

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

Example 7-1 Using GET_FACTOR to Retrieve a Factor

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

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

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

If you had set the DBMS_MACADM.CREATE_FACTOR or DBMS_MACADM.UPDATE_FACTOR eval_options parameter to factor evaluation to DBMS_MACUTL.G_EVAL_ON_SESSION, then Oracle Database Vault retrieves the value from the session context established, as described under How Factors Are Processed When a Session Is Established.

If you had set the factor evaluation to DBMS_MACUTL.G_EVAL_ON_ACCESS, then Oracle Database Vault performs Step 2 through Step 5 (or Step 6), as described under How Factors Are Processed When a Session Is Established, whenever the factor is retrieved.

If you had defined error options for the factor and if an error occurs, then Oracle Database Vault displays the error message.

7.7.4 How Factors Are Set

You can assign a factor identity at any time during a database session, but only if the factor assignment rule set evaluates to true.

You can do this in the application code by using the SET_FACTOR function. In Java code, you can use the JDBC class java.sql.CallableStatement to set this value. For example:

```
java.sql.Connection connection ;
...
java.sql.CallableStatement statement =
   connection.prepareCall("{call SET_FACTOR('FACTOR_X', ?)}");
statement.setString(1, "MyValue");
boolean result = statement.execute();
...
```

Applications that can execute Oracle PL/SQL functions can use this procedure (for example, applications written using Oracle Data Provider for .NET (ODP.NET)).

This concept is similar to the standard Oracle `DBMS_SESSION.SET_IDENTIFIER` procedure with an added feature that a rule set controls when a factor value can be set. If the rule set evaluates to true, Steps 2 through 5 under How Factors Are Processed When a Session Is Established occur.

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

7.7.5 How Factor Auditing Works

Whether you have unified auditing enabled affects how auditing is handled for factors.

In a traditional, non-unified auditing environment, Oracle Database Vault writes the audit trail to the `DVSYS.AUDIT_TRAIL$` table. Be aware that traditional auditing is desupported starting with Oracle Database 23c.

If you have enabled unified auditing, then this setting does not capture audit records. Instead, you can create unified audit policies to capture this information.

You can use the Factor Audit Report to display the generated audit records. 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.

Related Topics

- *Oracle Database Security Guide*
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

7.8 Tutorial: Preventing Ad Hoc Tool Access to the Database

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

- About This Tutorial
  Many database applications contain features to explicitly control the actions of a user.
- Step 1: Enable the HR User Accounts
  You must use the `HR` account later on when you test the Oracle Database Vault components for this tutorial.
- Step 2: Create the Factor
  After you have ensured that the `HR` account is active, you can create a factor.
- Step 3: Create the Rule Set and Rule
  After you have created the factor, you can create a rule set and a rule to work with the factor.
- Step 4: Create the CONNECT Command Rule
  The CONNECT command rule controls the `CONNECT` SQL statement.
- Step 5: Test the Ad Hoc Tool Access Restriction
  You do not need to restart your SQL*Plus session for the Oracle Database Vault changes to take effect.

- Step 6: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

7.8.1 About This Tutorial

Many database applications contain features to explicitly control the actions of a user.

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

In the following tutorial, you prevent user `HR` from using SQL*Plus. To accomplish this, you will create an Oracle Database Vault rule that uses the `CLIENT_PROGRAM_NAME` attribute of the `SYS_CONTEXT` SQL function `USERENV` namespace to find the names of the applications that are used to access the current instance of Oracle Database. Then you create a rule set that uses the new rule and a command rule for the `CONNECT` SQL statement, which is associated with the rule set.

Related Topics

- *Oracle Database SQL Language Reference*

7.8.2 Step 1: Enable the HR User Accounts

You must use the `HR` account later on when you test the Oracle Database Vault components for this tutorial.

1. Log into the PDB as a user who has been granted the `DV_ACCTMGR` role.
2. Check the status of the `HR` account.

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

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

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

   Replace *`password`* with a password that meets the password complexity requirements of the user's profile.

Related Topics

- *Oracle Database Security Guide*

7.8.3 Step 2: Create the Factor

After you have ensured that the `HR` account is active, you can create a factor.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Combine the `SYS_CONTEXT` function with the function `regexp_like` to create a flexible rule to identify SQL*Plus anywhere in the `CLIENT_PROGRAM_NAME` attribute.

The following query shows the output of the `CLIENT_PROGRAM_NAME` attribute and a value of `1` if the client used is SQL*Plus:

```
column client_program_name format a40
SELECT SYS_CONTEXT('USERENV', 'CLIENT_PROGRAM_NAME')
CLIENT_PROGRAM_NAME, REGEXP_LIKE(SYS_CONTEXT('USERENV',
'CLIENT_PROGRAM_NAME') ,'sqlplus','i')
USING_SQLPLUS FROM DUAL;
```

The output should be similar to the following:

```
CLIENT_PROGRAM_NAME                      USING_SQLPLUS
---------------------------------------- -------------
sqlplus@nemosity (TNS V1-V3)             TRUE
```

The following query will show the output of the `CLIENT_PROGRAM_NAME` attribute and a value of `0` if the client is not SQL*Plus. Notice the only code that has been changed in the query is the addition of the word `NOT` before the `REGEX_LIKE` function and the column alias is now `not_using_sqlplus` instead of `USING_SQLPLUS`.

```
SELECT SYS_CONTEXT('USERENV', 'CLIENT_PROGRAM_NAME')
CLIENT_PROGRAM_NAME, NOT REGEXP_LIKE(SYS_CONTEXT('USERENV',
'CLIENT_PROGRAM_NAME') ,'sqlplus','i') NOT_USING_SQLPLUS FROM DUAL;
```

Output similar to the following appears:

```
CLIENT_PROGRAM_NAME                      NOT_USING_SQLPLUS
---------------------------------------- -----------------
sqlplus@nemosity (TNS V1-V3)             FALSE
```

3. Create the factor.

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'is_sqlplus',
  factor_type_name  => 'Application',
  description       => 'Returns TRUE if the client program name that
connects to the database contains sqlplus',
  rule_set_name     => NULL,
  validate_expr     => NULL,
  get_expr          => 'REGEXP_LIKE(SYS_CONTEXT(''USERENV'',
''CLIENT_PROGRAM_NAME'') ,''sqlplus'',''i'')',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_METHOD,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_SESSION,
  audit_options     => DBMS_MACUTL.G_AUDIT_OFF,
  fail_options      => DBMS_MACUTL.G_FAIL_SILENTLY);
END;
/
```

In this specification:

- `factor_type_name` specifies that this is an application-based factor.

- `get_expr` defines the expression for the factor. This expression calls the `SYS_CONTEXT` function, using the `USERENV` namespace and `CLIENT_PROGRAM_NAME` attribute, to find the programs that are logged into the Oracle database.
- `identify_by` identifies the factor by method.
- `labeled_by` labels the identities for the factor directly from the labels associated with an Oracle Label Security policy (default).
- `eval_options` evaluates the factor when the database session is created.
- `audit_options` is disabled; audit records are written to the unified audit trail, viewable with the `UNIFIED_AUDIT_TRAIL` data dictionary view.
- `fail_silently` does not show any error messages for the factor.

7.8.4 Step 3: Create the Rule Set and Rule

After you have created the factor, you can create a rule set and a rule to work with the factor.

1. Create the Limit SQL*Plus Access rule set as follows:

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Limit SQL*Plus Access',
  description      => 'Limits access to SQL*Plus for Apps Schemas',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'SQL*Plus access not allowed for Apps Schemas',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
  handler          => NULL,
  is_static        => FALSE);
END;
/
```

   In this specification:

   - `fail_message` is a customized error message that will be returned to the user if the rule set criteria is not met.
   - `fail_options` enables an error message, set by `fail_message`, and error code, set by `fail_code`, to appear if there are errors.
   - `is_static` evaluates the rule set once during the user session. After that, the value is re-used.

2. Check the new factor.

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

   The output should be similar to the following:

```
F$IS_SQLPLUS
----------------------------------------
TRUE
```

3. Create the following rule.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
```

```
  rule_name    => 'Prevent HR Schema Access to SQL*Plus',
  rule_expr    => 'DVF.F$IS_SQLPLUS = ''FALSE''');
END;
/
```

4. Add the rule to the `Limit SQL*Plus Access` rule set.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit SQL*Plus Access',
  rule_name     => 'Prevent HR Schema Access to SQL*Plus',
  rule_order    => 1);
END;
/
```

The `rule_order` setting is required to enable the procedure to work.

7.8.5 Step 4: Create the CONNECT Command Rule

The CONNECT command rule controls the `CONNECT` SQL statement.

This command rule also applies to logging into SQL*Plus from the command line or other tools your site may use to access SQL*Plus.

- Create the CONNECT command rule as follows:

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'CONNECT',
  rule_set_name   => 'Limit SQL*Plus Access',
  object_owner    => 'HR',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

In this specification:

- `rule_set_name` associates the Limit SQL*Plus Access rule set with the CONNECT command rule.
- `object_owner` is set to `HR` so that the command rule applies only to `HR`.
- `object_name` is set to `%` so that the command rule applies to all objects.
- `enabled` enables the command rule so that it can be used right away.

7.8.6 Step 5: Test the Ad Hoc Tool Access Restriction

You do not need to restart your SQL*Plus session for the Oracle Database Vault changes to take effect.

1. In SQL*Plus, try to connect to the PDB as user `HR`:

```
CONNECT HR@pdb_name
Enter password: password
```

The following output should appear:

```
ERROR:
ORA-47306: 20461: SQL*Plus access not allowed for Apps Schemas
```

User HR should be prevented from using SQL*Plus.

2. Now try to connect as user SYSTEM:

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

User SYSTEM should be able to log in to the database instance. So should SYS, the Database Vault Owner account, and the Database Vault Account Manager account.

If the Test Fails

If you cannot log in to the database instance as SYSTEM (or as any of the other administrative users listed in your rule expression), then you are prevented from using SQL*Plus. Users with the DV_OWNER role are exempt from Oracle Database Vault connect command rules.

You can remedy the problem as follows:

1. Log in to the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. Enter the following statement to drop the CONNECT command rule.

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

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

3. Check the policy components for any errors and then correct them. Recreate the CONNECT command rule, and then test it.

7.8.7 Step 6: Remove the Components for This Tutorial

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

1. Connect to the PDB as the user who has been granted the DV_OWNER or DV_ADMIN role.
2. Remove the CONNECT command rule.

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

3. Remove the Limit SQL*Plus Access rule set.

```
EXEC DBMS_MACADM.DELETE_RULE_SET('Limit SQL*Plus Access');
```

4. Remove the rule.

```
EXEC DBMS_MACADM.DELETE_RULE('Prevent HR Schema Access to SQL*Plus');
```

5. Remove the Client_Prog_Name factor.

```
EXEC DBMS_MACADM.DELETE_FACTOR('IS_SQLPLUS');
```

6. If necessary, as a user who has been granted the DBV_ACCTMGR role, lock the HR account.

```
ALTER USER HR ACCOUNT LOCK;
```

7.9 Guidelines for Designing Factors

Oracle provides guidelines for designing factors.

- You can use the Oracle utility packages such as `UTL_TCP`, `UTL_HTTP`, `DBMS_LDAP`, and `DBMS_PIPE` to integrate security or other contextual information about the session from external systems.
- Do not specify a retrieval method (using the `get_expr` parameter in `DBMS_MACADM.CREATE_FACTOR` or `DBMS_MACADM.UPDATE_FACTOR`) if the `identify_by` parameter is set to `DBMS_MACUTL.G_IDENTIFY_BY_FACTOR`. Retrieval methods are only needed if you set the factor to `DBMS_MACUTL.G_IDENTIFY_BY_CONSTANT` or `DBMS_MACUTL.G_IDENTIFY_BY_METHOD`.
- 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 `eval_options`) of `DBMS_MACUTL.G_EVAL_ON_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.
- 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.
- It is generally easier to configure and debug a factor whose `labeled_by` parameter is set to `DBMS_MACUTL.G_LABELED_BY_SELF` than one labeled `DBMS_MACUTL.G_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.

Related Topics

- Integrating Oracle Database Vault with Oracle Label Security
  You can integrate Oracle Database Vault with Oracle Label Security, and check the integration with reports and data dictionary views.

7.10 How Factors Affect Performance

The complexity of factors affects the performance of your Oracle database instance.

Each factor has elements that are processed, such as its validation method, trust level, and so on. For factors that are evaluated by the session, such as Database_Hostname and `Proxy_User`, Oracle Database Vault performs this processing during session initialization, and then caches the results for subsequent requests for that value.

The default factors are cached because they are likely candidates for a typical security policy. However, if you only use five factors (for example, in rule sets or other components), then the other factors consume resources that could otherwise be used elsewhere. In this case, you should remove the unnecessary factors by deleting them. (Oracle Database Vault does not use any of these factors internally, so you can remove them if you do not need them.)

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

You can check system performance by running tools such as Oracle Enterprise Manager (including Oracle Enterprise Manager Cloud Control, which is installed by default with Oracle Database), Automatic Workload Repository (AWR), and `TKPROF`.

Related Topics

- Default Factors
  Oracle Database Vault provides a set of default factors.
- *Oracle Database Performance Tuning Guide*
- *Oracle Database SQL Tuning Guide*

7.11 Factor Related Reports and Data Dictionary Views

Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and data dictionary views that are useful for analyzing factors and their identities.

Table 7-1 lists the Oracle Database Vault reports.

Table 7-1 Oracle Enterprise Manager Reports Related to Factors and Their Identities

Report	Description
Factor Audit Report	Audits factors (for example, to find factors that failed to be evaluated)
Factor Configuration Issues Report	Lists configuration issues, such as disabled or incomplete rule sets, or to audit issues that may affect the factor
Factor Without Identities Report	Lists factors that have had no identities assigned yet
Identity Configuration Issues Report	Lists factors that have invalid label identities or no map for the identity

Table 7-1 (Cont.) Oracle Enterprise Manager Reports Related to Factors and Their Identities

Report	Description
Rule Set Configuration Issues Report	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	Lists the existing factors in the current database instance
DBA_DV_FACTOR_LINK	Shows the relationships of each factor whose identity is determined by the association of child factors
DBA_DV_FACTOR_TYPE	Lists the names and descriptions of factor types used in the system
DBA_DV_IDENTITY	Lists the identities for each factor
DBA_DV_IDENTITY_MAP	Lists the mappings for each factor identity

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

8

Configuring Secure Application Roles for Oracle Database Vault

Secure application roles enable you to control how much access users have to an application.

- What Are Secure Application Roles in Oracle Database Vault?
  In Oracle Database Vault, you can create a secure application role that you enable with an Oracle Database Vault rule set.
- Security for Oracle Database Vault Secure Application Roles
  Users who have database administrative privileges may try to use the `DROP ROLE` statement to delete Oracle Database Vault secure application roles.
- Creating an Oracle Database Vault Secure Application Role
  When you create a secure application role, you associate it with a rule set to determine when the role is enabled or disabled.
- Enabling Oracle Database Secure Application Roles to Work with Oracle Database Vault
  You can modify an existing secure application role only if it has been created in Oracle Database Vault.
- Modifying a Secure Application Role
  You can modify the definition of an Oracle Database Vault secure application role.
- Deleting an Oracle Database Vault Secure Application Role
  You can delete Oracle Database Vault secure application roles if no applications are using them.
- How Oracle Database Vault Secure Application Roles Work
  The process flow for an Oracle Database Vault secure application role begins after you create and set the secure application role.
- Tutorial: Granting Access with Database Vault Secure Application Roles
  This tutorial demonstrates how to create a secure application role to control user access to the `OE.ORDERS` table during work hours.
- How Secure Application Roles Affect Performance
  You can check system performance by using Oracle Database tools, including Oracle Enterprise Manager Cloud Control.
- Secure Application Role Related Reports and Data Dictionary View
  Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and a data dictionary view that are useful for analyzing Oracle Database Vault secure application roles.

8.1 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.

You only can create a secure application role in a PDB, not in the CDB root or the application root.

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

Related Topics

- Oracle Database Vault Secure Application Role APIs
  The `DBMS_MACADM` and `DBMS_MACSEC_ROLES` PL/SQL packages manage Database Vault secure application roles.

8.2 Security for Oracle Database Vault Secure Application Roles

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

Whenever an Oracle Database Vault secure application role has been created, Database Vault adds the secure application role to the Oracle Database Vault realm. This prevents database administrator from deleting the secure application role using the `DROP ROLE` statement.

8.3 Creating an Oracle Database Vault Secure Application Role

When you create a secure application role, you associate it with a rule set to determine when the role is enabled or disabled.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. If necessary, create a rule set that the secure application role will use.

   The `DBA_DV_RULE_SET` data dictionary view lists existing rule sets.
3. Run the `DBMS_MACADM.CREATE_ROLE` procedure to create the security role.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_ROLE(
  role_name      => 'access_hr_employees',
  enabled        => DBMS_MACUTL.G_YES,
```

```
  rule_set_name  => 'Can Access HR.EMPLOYEES');
END;
/
```

In this specification:

- `role_name` can be up to 128 characters in mixed-case, without spaces. Ensure that this name follows the standard Oracle naming conventions for role creation using the `CREATE ROLE` statement. The `DBA_DV_FACTOR` data dictionary view lists existing factors. This parameter is mandatory. The `DBA_DV_ROLE` data dictionary view lists existing security roles.
- `enabled` enables or disables the role to be available for use. This parameter is mandatory. `DBMS_MACUTL.G_YES` makes the role available for enabling; `DBMS_MACUTL.G_NO` prevents the role from being enabled. The default is `DBMS_MACUTL.G_YES`. That is, users are allowed to call the `DBMS_MACSEC_ROLES.SET_ROLE` function to try to enable the role. Note that whether or not the role will be enabled depends on the evaluation result of the associated rule set.
- `rule_set_name` is a mandatory rule set that the `DBMS_MACSEC_ROLES.SET_ROLE` procedure will use to determine if the role should be enabled or disabled. If the rule set evaluates to true, then Oracle Database Vault enables the role for the database session. If the rule set evaluates to false, then the role is not enabled. The `DBA_DV_RULE_SET` data dictionary view lists existing rule sets.

4. As the owner of the schema that will be affected by the secure application role, grant the appropriate privileges to the secure application role.

   These privileges should be the same privileges that the secure application role will control. For example, suppose you created a role that enabled users to select or update the `HR.EMPLOYEES` table. The `HR` user would need to grant the `SELECT` and `UPDATE` privileges to the secure application role.

   For example:

   ```
   CONNECT HR@pdb_name
   Enter password: password

   GRANT SELECT, UPDATE ON EMPLOYEES TO ACCESS_HR_EMPLOYEES;
   ```

5. Test the secure application role.

   a. Connect as the user who will be granted or denied the secure application role.

   b. Run the `DBMS_MACSEC_ROLES.SET_ROLE` procedure on the role. For example:

      ```
      EXEC DBMS_MACSEC_ROLES.SET_ROLE('ACCESS_HR_EMPLOYEES');
      ```

   c. Attempt to perform an action that is controlled by the secure application role. For example:

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

      If the user should be granted the privileges, then the user can perform the action. Otherwise, the action will fail.

Related Topics

- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.

- SET_ROLE Procedure
  The `SET_ROLE` procedure issues the `SET ROLE` PL/SQL statement for specified roles.
- Oracle Database Vault Secure Application Role APIs
  The `DBMS_MACADM` and `DBMS_MACSEC_ROLES` PL/SQL packages manage Database Vault secure application roles.

8.4 Enabling Oracle Database Secure Application Roles to Work with Oracle Database Vault

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. However, you can enable non-Oracle Database Vault roles to work with Oracle Database Vault.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Create a new secure application role in Oracle Database Vault and then grant the existing role to the secure application role.

   For example:

   ```
   GRANT myExistingDBrole TO myDVrole;
   ```

3. Modify your code to use this new role.

   You can use `DBMS_MACSEC_ROLES.SET_ROLE` in your application code to accomplish this.

Related Topics

- SET_ROLE Procedure
  The `SET_ROLE` procedure issues the `SET ROLE` PL/SQL statement for specified roles.

8.5 Modifying a Secure Application Role

You can modify the definition of an Oracle Database Vault secure application role.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Find the secure application role and check its definition.

   For example:

   ```
   SELECT * FROM DBA_DV_ROLE ORDER BY ROLE;
   ```

3. Run the `DBMS_MACADM.UPDATE_ROLE` statement.

   For example:

   ```
   BEGIN
    DBMS_MACADM.UPDATE_ROLE(
     role_name      => 'access_hr_employees',
     enabled        => DBMS_MACUTL.G_NO,
   ```

```
  rule_set_name  => 'System Access Controls');
END;
/
```

Related Topics

- Oracle Database Vault Secure Application Role APIs
  The `DBMS_MACADM` and `DBMS_MACSEC_ROLES` PL/SQL packages manage Database Vault secure application roles.

8.6 Deleting an Oracle Database Vault Secure Application Role

You can delete Oracle Database Vault secure application roles if no applications are using them.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_ROLE` data dictionary view to find the secure application roles that you want to delete.

   For example:

   ```
   SELECT ROLE FROM DBA_DV_ROLE ORDER BY ROLE;
   ```

3. Check and modify any applications that may be using the secure application role that you want to delete.
4. Run the `DBMS_MACADM.DELETE_ROLE` procedure to delete the role.

   For example:

   ```
   EXEC DBMS_MACADM.DELETE_ROLE('access_hr_employees');
   ```

Related Topics

- Oracle Database Vault Secure Application Role APIs
  The `DBMS_MACADM` and `DBMS_MACSEC_ROLES` PL/SQL packages manage Database Vault secure application roles.

8.7 How Oracle Database Vault Secure Application Roles Work

The process flow for an Oracle Database Vault secure application role begins after you create and set the secure application role.

1. Create or update the role either in Oracle Database Vault Administrator or by using the secure application role-specific functions in the `DBMS_MACADM` package.
2. Modify your application to call the role, by using the `DBMS_MACSEC_ROLES.SET_ROLE` function.
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.

Related Topics

- DBMS_MACADM Secure Application Role Procedures
  The `DBMS_MACADM` package creates, renames, assigns, unassigns, updates, and deletes Oracle Database Vault secure application roles.
- SET_ROLE Procedure
  The `SET_ROLE` procedure issues the `SET ROLE` PL/SQL statement for specified roles.

8.8 Tutorial: Granting Access with Database Vault Secure Application Roles

This tutorial demonstrates how to create a secure application role to control user access to the `OE.ORDERS` table during work hours.

- About This Tutorial
  In this tutorial, you restrict the `SELECT` statement on the `ORDERS` table in the `OE` schema to a specific set of users.
- Step 1: Create Users for This Tutorial
  First, you must create users for the tutorial.
- Step 2: Enable the OE User Account
  The `OE` schema will be used for this tutorial.
- Step 3: Create the Rule Set and Its Rules
  The rule set and rules will restrict who can modify orders in the `OE.ORDERS` table.
- Step 4: Create the Database Vault Secure Application Role
  The Database Vault secure application role will be set when the rule set conditions are satisfied.
- Step 5: Grant the SELECT Privilege to the Secure Application Role
  The secure application role must be granted the `SELECT` privilege.
- Step 6: Test the Database Vault Secure Application Role
  With all the components in place, you can test the Database Vault secure application role.
- Step 7: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

8.8.1 About This Tutorial

In this tutorial, you restrict the `SELECT` statement on the `ORDERS` table in the `OE` schema to a specific set of users.

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

8.8.2 Step 1: Create Users for This Tutorial

First, you must create users for the tutorial.

1. Log in to a PDB as a user who has been granted the `DV_ACCTMGR` role.
2. Create the following user accounts:

```
GRANT CREATE SESSION TO eabel IDENTIFIED BY password;
GRANT CREATE SESSION TO ahutton IDENTIFIED BY password;
GRANT CREATE SESSION TO ldoran IDENTIFIED BY password;
```

   Replace *password* with a password that meets the password complexity requirements of the user's profile.

Related Topics

- *Oracle Database Security Guide*

8.8.3 Step 2: Enable the OE User Account

The `OE` schema will be used for this tutorial.

1. In SQL*Plus, connect as the `DV_ACCTMGR` user.
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;
```

   Replace *password* with a password that meets the password complexity requirements of the user's profile.

Related Topics

- *Oracle Database Security Guide*

8.8.4 Step 3: Create the Rule Set and Its Rules

The rule set and rules will restrict who can modify orders in the `OE.ORDERS` table.

1. Connect as a user who has been granted the `DV_OWNER` role.
2. Create the following rule set.

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Can Modify Orders',
  description      => 'Rule set to control who can modify orders in the OE.ORDERS
table',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'Failure',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
```

```
  handler          => '',
  is_static        => FALSE,
  scope            => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

3. Create the following rule.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Check Session User',
  rule_expr  =>'DVF.F$SESSION_USER IN (''EABEL'',''AHUTTON'')');
END;
/
```

4. Add the `Check Session User` rule to the `Can Modify Orders` rule set.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Can Modify Orders',
  rule_name     => 'Check Session User',
  rule_order    => 1);
END;
/
```

8.8.5 Step 4: Create the Database Vault Secure Application Role

The Database Vault secure application role will be set when the rule set conditions are satisfied.

1. If necessary, connect as the user who was granted the `DV_OWNER` role.
2. Create and enable the secure application role, and associate it with the `Can Modify Orders` rule set.

```
BEGIN
 DBMS_MACADM.CREATE_ROLE(
  role_name      => 'ORDERS_MGMT',
  enabled        => DBMS_MACUTL.G_YES,
  rule_set_name  => 'Can Modify Orders');
END;
/
```

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.

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

The secure application role must be granted the `SELECT` privilege.

1. In SQL*Plus, connect as user `OE`.
2. Grant the `SELECT` privilege to the `ORDERS_MGMT` secure application role.

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

8.8.7 Step 6: Test the Database Vault Secure Application Role

With all the components in place, you can test the Database Vault secure application role.

1. Connect as user `eabel`.
2. Set the `ORDERS_MGMT` role.

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

   Typically, you would embed this call in the application to which the user logs in.
3. Select from the `OE.ORDERS` table.

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

   The following output should appear:

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

   Because user `eabel` is configured 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. Connect as user `ldoran`.
5. Enter the following statements:

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

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

8.8.8 Step 7: Remove the Components for This Tutorial

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

1. Connect as a user who was granted the `DV_OWNER` role.
2. Drop the `ORDERS_MGMT` secure application role.

   ```
   EXEC DBMS_MACADM.DELETE_ROLE('ORDERS_MGMT');
   ```
3. Remove the `Check Session User` rule from the `Can Modify Orders` rule set.

   ```
   BEGIN
    DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
     rule_set_name => 'Can Modify Orders',
     rule_name     => 'Check Session User');
   END;
   /
   ```
4. Drop the rule and rule set.

   ```
   EXEC DBMS_MACADM.DELETE_RULE('Check Session User');
   EXEC DBMS_MACADM.DELETE_RULE_SET('Can Modify Orders');
   ```
5. Connect as a user who has been granted the `DV_ACCTMGR` role.

6. Drop the users.

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

7. If unnecessary, lock and expire the `OE` user account.

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

8.9 How Secure Application Roles Affect Performance

You can check system performance by using Oracle Database tools, including Oracle Enterprise Manager Cloud Control.

Other tools that you can use are Automatic Workload Repository (AWR) and `TKPROF`.

Related Topics

- *Oracle Database Performance Tuning Guide*
- *Oracle Database SQL Tuning Guide*

8.10 Secure Application Role Related Reports and Data Dictionary View

Together, Oracle Enterprise Manager and Oracle Database Vault provide reports and a data dictionary view that are useful for analyzing Oracle Database Vault secure application roles.

Table 8-1 lists the Oracle Database Vault reports.

Table 8-1 Oracle Enterprise Manager Reports Related to Secure Application Roles

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

The `DBA_DV_ROLE` data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

9

Configuring Oracle Database Vault Policies

You can use Oracle Database Vault policies to implement frequently used realm and command rule settings.

- What Are Database Vault Policies?
  An Oracle Database Vault policy groups local realms and command rules into a named policy that you can enable or disable as necessary.
- Default Oracle Database Vault Policies
  Oracle Database Vault provides two default policies that you can use to better secure user accounts and system privileges.
- Creating an Oracle Database Policy
  To create an Oracle Database Vault policy, you create a container policy that specifies the realms and command rules that encompass the policy.
- Modifying an Oracle Database Vault Policy
  You can use the modify an Oracle Database Vault policy.
- Deleting an Oracle Database Vault Policy
  You can use Enterprise Manager Cloud Control to delete Oracle Database Vault policies.
- Oracle Database Vault Policies Data Dictionary Views
  Oracle Database Vault provides data dictionary views that are useful for analyzing Database Vault policies.

9.1 What Are Database Vault Policies?

An Oracle Database Vault policy groups local realms and command rules into a named policy that you can enable or disable as necessary.

- About Oracle Database Vault Policies
  Oracle Database Vault policies can group realm and command rule definitions into one policy, which then can be collectively enabled or disabled.
- Oracle Database Vault Policies in a Multitenant Environment
  Oracle Database Vault policies are only local to the pluggable database (PDB) in which they were created.

9.1.1 About Oracle Database Vault Policies

Oracle Database Vault policies can group realm and command rule definitions into one policy, which then can be collectively enabled or disabled.

Database Vault policies enable you to delegate limited realm administration privileges to database users without giving them the powerful privileges that the `DVADM` and `DVOWNER` roles provide. Oracle Database Vault provides default policies.

For example, suppose you have a set of Oracle Database Vault objects that are related to a particular application, such as a realm and several command rules. You can use a Database Vault policy to group these objects into one policy. You then can designate a policy administrator to manage adding users to a realm for this application and for enabling or

disabling the policy. If there is only one primary application, then it can be used for manageability where a user can enable, disable, or simulate (use simulation mode) all related objects with one command rather than issuing a command for each included Database Vault object.

How the enablement of the individual realms and command rules works depends on how you set the policy state of the policy, as follows:

- Full enabled mode (`DBMS_MACADM.G_ENABLED`) sets the policy to take precedence over the individual enablement settings of the associated realms and command rules. For example, if the associated objects of a policy are individually disabled, then they will be enabled if the policy is enabled. (Conversely, you can set `DBMS_MACADM.G_PARTIAL` to allow the embedded security objects to set their own enabled, disabled, or simulation mode.)
- Partial enabled mode (`DBMS_MACADM.G_PARTIAL`) enables the associated realms and command rules to have different status settings (`ENABLED`, `DISABLED`, and `SIMULATION`). The other policy status choices force all associated controls to the same status dictated by the policy. Setting the policy status to partial allows each realm and command rule to change status as required.
- Simulation mode (`DBMS_MACACM.G.SIMULATION`) enables the policy but writes violations to realms or command rules to a designated log table with information about the type of violation, such as a user name or the SQL statement that was used. Simulation forces every security object in the policy to be in simulation mode.
- Disabled mode (`DBMS_MACADM.G_DISABLED`) disables the policy after you create it.

In general, to create a Database Vault policy, you perform the following steps:

1. Create the necessary realms and command rules to use in the policy.
2. Create the Database Vault policy.

   You can use the `DBMS_MACADM.CREATE_POLICY` procedure to create the policy.
3. Add one or more realms to the policy.

   You can use the `DBMS_MACADM.ADD_REALM_TO_POLICY` procedure to add realms to the policy.
4. Add one or more command rules to the policy.

   You can use the `DBMS_MACADM.ADD_CMD_TO_POLICY` procedure to add command rules to the policy.
5. Add one or more database users as owners of the policy.

   You can use the `DBMS_MACADM.ADD_OWNER_TO_POLICY` procedure to add users to the policy. Afterward, grant this user the `DV_POLICY_OWNER` role. This user will be able to perform a limited set of tasks: changing the policy state, adding or removing authorization from a realm, and having the `SELECT` privilege for a set of the `DVSYS.POLICY_OWNER*` data dictionary views. By default, the `DVOWNER` user owns the policy.

After the policy is created, it can be used right away.

Related Topics

- Default Oracle Database Vault Policies
  Oracle Database Vault provides two default policies that you can use to better secure user accounts and system privileges.
- Oracle Database Vault Policy APIs
  You can use the `DBMS_MACADM` PL/SQL package to manage Oracle Database Vault policies.
- DV_POLICY_OWNER Database Vault Owner Role
  The `DV_POLICY_OWNER` role enables database users to manage to a limited degree Oracle Database Vault policies.

9.1.2 Oracle Database Vault Policies in a Multitenant Environment

Oracle Database Vault policies are only local to the pluggable database (PDB) in which they were created.

That is, if you created the policy in a PDB, then only local realms and command rules can be added to it.

9.2 Default Oracle Database Vault Policies

Oracle Database Vault provides two default policies that you can use to better secure user accounts and system privileges.

You can use the default policies in your own security configurations. If you do not need them, then you can remove them because they are not needed for internal use by Oracle Database Vault.

The default policies are as follows:

- `Oracle Account Management Controls` enforces controls over user-related operations within Oracle Database Vault. It is used to prevent ad hoc user account creation, user deletions, and other user account-related operations by unauthorized privileged users. It includes the `Database Vault Account Management` realm and user account management command rules for SQL statements such as `CREATE USER`.
- `Oracle System Protection Controls` enforces controls on important database schemas, privileges, and roles that are associated with the default Oracle Database environment. It includes the realms such as `Oracle Default Schema Protection Realm` and command rules for the system management SQL statement `ALTER SYSTEM`.

Related Topics

- DBA_DV_POLICY_OBJECT View
  The `DBA_DV_POLICY_OBJECT` data dictionary view lists information about the objects that are protected by Oracle Database Vault policies in the current database instance.

9.3 Creating an Oracle Database Policy

To create an Oracle Database Vault policy, you create a container policy that specifies the realms and command rules that encompass the policy.

You can enable the policy during creation time, or enable it later on by executing the .

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Run the `DBMS_MACADM.CREATE_POLICY` procedure to create the policy

   For example:

   ```
   BEGIN
    DBMS_MACADM.CREATE_POLICY(
     policy_name  => 'OE Policy',
     description  => 'Policy to protect the OE schema',
     policy_state => DBMS_MACADM.G_ENABLED,
     pl_sql_stack => TRUE);
   END;
   /
   ```

   In this specification:

   - `policy_name` can be up to 128 characters in mixed case. The `DBA_DV_POLICY` data dictionary view lists existing policies.
   - `description` can be up to 4000 characters in mixed-case.
   - `policy_state` enables or disables the policy, using one of the following settings:
     - `DBMS_MACADM.G_ENABLED` (or `1`) enables the policy after you create it.
     - `DBMS_MACADM.G_DISABLED` (or `0`) disables the policy after you create it.
     - `DBMS_MACADM.G_SIMULATION` (or `2`) sets the policy to simulation mode. In simulation mode, any violations to realms or command rules used in the policy are logged in a designated log table with sufficient information to describe the error, such as the user name or SQL statement used.
     - `DBMS_MACADM.G_PARTIAL` (or `3`) sets the policy to partial mode. In partial mode, the enforcement state of realms or command rules associated with the policy can be changed individually.
   - `pl_sql_stack` is used when simulation mode is enabled and specifies whether to record the PL/SQL stack for failed operations. Enter `TRUE` to record the PL/SQL stack, `FALSE` to not record.
3. So that the Database Vault policy owner can query policy related views and run the allowed procedures, grant this user the `DV_POLICY_OWNER` role.

   You can grant this role to multiple users.

   For example:

   ```
   GRANT DV_POLICY_OWNER TO psmith, pfitch;
   ```
4. To add a database user as the owner of the policy, run the `DBMS_MACADM.ADD_OWNER_TO_POLICY` procedure.

   The policy owner will be able to modify the policy.

   For example:

   ```
   BEGIN
    DBMS_MACADM.ADD_OWNER_TO_POLICY(
     policy_name    => 'OE Policy',
     owner_name     => 'PSMITH');
   END;
   /
   ```

5. To add a command rule to the policy, run the `DBMS_MACADM.ADD_CMD_RULE_TO_POLICY` procedure.

   If you created the policy in a PDB, then the command rule must be local to this PDB.

   For example, for a simple command rule:

```
BEGIN
 DBMS_MACADM.ADD_CMD_RULE_TO_POLICY(
  policy_name    => 'OE Policy',
  command        => 'SELECT',
  object_owner   => 'OE',
  object_name    => 'ORDERS',
  scope          => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

   In this specification, the command rule must exist and match the parameters included. To fine the command rule definition, query the `DBA_DV_COMMAND_RULE`.

   If you want to add an `ALTER SYSTEM` or `ALTER SESSION` command rule, then you must include the parameters specific to those command rules. For example:

```
BEGIN
 DBMS_MACADM.ADD_CMD_RULE_TO_POLICY(
  policy_name    => 'OE Policy',
  command        => 'ALTER SESSION',
  object_owner   => '%',
  object_name    => '%',
  clause_name    => 'PARALLEL DDL',
  parameter_name => '',
  event_name     => '',
  action_name    => '',
  scope          => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

6. To add a realm to the policy, run the `DBMS_MACADM.ADD_REALM_TO_POLICY` procedure.

   If you created the policy in a PDB, then the command rule must be local to this PDB.

   For example:

```
BEGIN
 DBMS_MACADM.ADD_REALM_TO_POLICY(
  policy_name    => 'OE Policy',
  realm_name     => 'Database Vault Account Management');
END;
/
```

   In this specification:

   - `policy_name` is a name of the policy. The `DBA_DV_POLICY` view lists existing policies.
   - `realm_name` is the name of the realm. The `DBA_DV_REALM` view lists existing realms.

Related Topics

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

- Oracle Database Vault Policy APIs
  You can use the DBMS_MACADM PL/SQL package to manage Oracle Database Vault policies.

9.4 Modifying an Oracle Database Vault Policy

You can use the modify an Oracle Database Vault policy.

You can modify only the description and state of a policy. If you want to make other modifications, such as changing the realm that is associated with the policy, then you must delete the object from the policy (for example, with the DBMS_MACADM.DELETE_REALM_FROM_POLICY procedure) and then add the replacement object (for example, with DBMS_MACADM.ADD_REALM_TO_POLICY) to the policy.

1. Connect to the PDB as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. Find policy and check its definition.

   For example:

   ```
   SELECT * FROM DBA_DV_POLICY ORDER BY NAME;
   ```

3. To change the policy description, run the DBMS_MACADM.UPDATE_POLICY_DESCRIPTION procedure.

   For example:

   ```
   BEGIN
    DBMS_MACADM.UPDATE_POLICY_DESCRIPTION(
     policy_name  => 'OE Policy',
     description  => 'Policy to protect the OE schema from external intruders');
   END;
   /
   ```

4. To change the policy state, run the DBMS_MACADM.UPDATE_POLICY_STATE procedure.

   For example:

   ```
   BEGIN
    DBMS_MACADM.UPDATE_POLICY_STATE(
     policy_name   => 'OE Policy',
     policy_state  => DBMS_MACADM.G_SIMULATION,
     pl_sql_stack  => TRUE);
   END;
   /
   ```

Related Topics

- Oracle Database Vault Policy APIs
  You can use the DBMS_MACADM PL/SQL package to manage Oracle Database Vault policies.

9.5 Deleting an Oracle Database Vault Policy

You can use Enterprise Manager Cloud Control to delete Oracle Database Vault policies.

When you delete an Oracle Database Vault policy, the underlying realms and command rules are preserved, and they retain their individual enablement status. You

do not need to remove any objects (such as realms) that are associated with the policy before deleting it.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_POLICY_OBJECT` data dictionary view to find the policy to delete.

   For example:

   ```
   SELECT POLICY_NAME FROM DBA_DV_POLICY ORDER BY POLICY_NAME;
   ```

3. Run the `DBMS_MACADM.DROP_POLICY` procedure to drop the policy.

   For example:

   ```
   EXEC DBMS_MACADM.DROP_POLICY ('OE Policy');
   ```

9.6 Oracle Database Vault Policies Data Dictionary Views

Oracle Database Vault provides data dictionary views that are useful for analyzing Database Vault policies.

Table 9-1 lists data dictionary views that provide information about existing Oracle Database Vault policies.

Table 9-1 Data Dictionary Views Used for Oracle Database Vault Policies

Data Dictionary View	Description
`DBA_DV_POLICY`	Lists the Database Vault policies, a description, and their state
`DBA_DV_POLICY_OBJECT`	Provides detailed information about the policies, such as the associated realms and command rules
`DBA_DV_POLICY_OWNER`	Lists the owners of Database Vault policies
`DBA_DV_REALM_AUTH`	Enables users who have been granted the `DV_POLICY_OWNER` role to find information about the authorization that was granted to realms that have been associated with Database Vault policies, such as the realm name, grantee, and associated rule set.
`DVSYS.POLICY_OWNER_COMMAND_RULE`	Enables users who have been granted the `DV_POLICY_OWNER` role to find information about the command rules that have been associated with Database Vault policies, such as the command rule name.
`DVSYS.POLICY_OWNER_POLICY`	Enables users who have been granted the `DV_POLICY_OWNER` role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners
`DVSYS.POLICY_OWNER_REALM`	Enables users who have been granted the `DV_POLICY_OWNER` role to find information about the realms that have been associated with Database Vault policies, such as the realm name, realm type, or traditional audit options

Table 9-1 (Cont.) Data Dictionary Views Used for Oracle Database Vault Policies

Data Dictionary View	Description
DVSYS.POLICY_OWNER_REALM_OBJECT	Enables users who have been granted the DV_POLICY_OWNER role to find information about the objects that have been added to realms that are associated with Database Vault policies, such as the realm name, grantee, and associated rule set
DVSYS.POLICY_OWNER_RULE	Enables users who have been granted the DV_POLICY_OWNER role to find information about the rules that have been associated with rule sets in Database Vault policies, such as the rule name and its expression
DVSYS.POLICY_OWNER_RULE_SET	Enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies, such as the rule set name, its handler information, and whether it is enabled
DVSYS.POLICY_OWNER_RULE_SET_RULE	Enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that contain rules used in Database Vault policies, such as the rule set name and whether it is enabled

Related Topics

- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

10

Using Simulation Mode for Logging Realm and Command Rule Activities

Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

- About Simulation Mode
  Simulation mode enables you to capture violations in a simulation log instead of blocking SQL execution by Oracle Database Vault realms and command rules.
- Simulation Mode Use Cases
  Simulation mode is useful for testing a development configuration of new realms and command rules.
- Logging Realms in Simulation Mode
  You can set both regular and mandatory realms in simulation mode.
- Tutorial: Tracking Violations to a Realm Using Simulation Mode
  This tutorial shows how to create a realm that uses simulation mode and then test violations to the realm.

10.1 About Simulation Mode

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

Simulation mode stores the errors that are captured in one location for easy analysis. To use simulation mode, when you create or update a realm or command rule, instead of enabling or disabling the realm or command rule, you can set it to simulation mode. The realm or command rule is still enabled, but because violations are not blocked and are instead recorded to the simulation log file, you can test it for any potential errors before you enable it for a production environment. When simulation mode is enabled, the report may include violations for multiple realms or command rules. For more detailed reports that can help you better identify the source of user SQL statements, you can configure simulation mode to include the PL/SQL call stack. The call stack captures the calling procedures and functions recursively to better troubleshoot the Database Vault. Call stack information is stored in the `PL_SQL_STACK` column in the `DVSYS.DBA_DV_SIMULATION_LOG` data dictionary view.

For example, the following creation statement for a realm enables simulation mode and generates the PL/SQL call stack:

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
 realm_name    => 'HR Apps',
 description   => 'Realm to protect the HR realm',
 enabled       => DBMS_MACUTL.G_SIMULATION,
 audit_options => DBMS_MACUTL.G_REALM_AUDIT_OFF,
 realm_type    => 1,
 realm_scope   => DBMS_MACUTL.G_SCOPE_LOCAL,
 pl_sql_stack  => TRUE);
END;
/
```

At this stage, SQL statements that violate realms or command rules are still able to execute, but these activities are recorded to the DBA_DV_SIMULATION_LOG data dictionary view. For example, the following query finds violations against the HR Apps realm and any other realms or command rules that have been configured for simulation mode:

```
SELECT USERNAME, COMMAND, SQLTEXT, VIOLATION_TYPE
FROM DBA_DV_SIMULATION_LOG
WHERE REALM_NAME = "HR APPS";

USERNAME  COMMAND    SQLTEXT                          VIOLATION_TYPE

--------  ---------- -------------------------------  --------------
DGRANT    SELECT     SELECT SALARY FROM HR.EMPLOYEES; Realm Violation
```

After you have completed testing the realm or command rule, a user who has been granted the DV_ADMIN or DV_OWNER role can clear the DBA_DV_SIMULATION_LOG data dictionary view by deleting the contents of the underlying table of this view, DVSYS.SIMULATION_LOG$.

For example:

```
DELETE FROM DVSYS.SIMULATION_LOG$;
```

Or:

```
DELETE FROM DVSYS.SIMULATION_LOG$ WHERE COMMAND = 'SELECT';
```

10.2 Simulation Mode Use Cases

Simulation mode is useful for testing a development configuration of new realms and command rules.

Use cases are as follows:

- Application certification

  When you are certifying applications, you can use simulation mode as follows in an application test environment:

  1. Put all schemas for the application into mandatory realms with simulation mode enabled.
  2. Run a full regression test.
  3. Analyze the simulation mode log by querying the DBA_DV_SIMULATION_LOG data dictionary view to find who can access these schemas.
  4. Update the realms with new authorizations, and the enable the realms (that is, not using simulation mode).
  5. Re-run the regression test.

- Introduction of a new command rule

  You can use simulation mode on a production database that has Oracle Database Vault enabled.

  1. Put the new command rule into production in simulation mode for however many weeks that are necessary.

2. Analyze the simulation mode log by querying `DBA_DV_SIMULATION_LOG` to determine if the command rule is working correctly.
3. Make changes to the command rule as necessary.
4. Enable the command rule.

- Putting a new realm into a production database in simulation mode.

  This method can help to find the system context information needed to set the trusted path rules in rule sets and find authorized users for realms.

  1. Create the new realm in mandatory mode and add the protected objects
  2. Do not add any authorized users.
  3. Run applications and development operations from the normal IP addresses that will be used.
  4. Check the simulation log file for both authorized users and system context information that you can use to create trusted paths.
  5. Create the trusted paths, and then add the authorized users.
  6. Clear the simulation log and run the application and development operation tasks again.
  7. After a period of time, review the simulation log. If all the controls were updated correctly, then the simulation log is empty. Log entries in the simulation mode indicate additional changes that you need to make to the realm and rule sets or the log entries may indicate a malicious use.

10.3 Logging Realms in Simulation Mode

You can set both regular and mandatory realms in simulation mode.

- Considerations When Logging Realms in Simulation Mode
  There are several use cases to consider if you want to use realms in simulation mode.
- Use Case: All New Realms in Simulation Mode
  In this use case, all realms are either mandatory or regular and all user-created realms are in simulation mode.
- Use Case: New Realms Introduced to Existing Realms
  In this use case, you add a set of new realms to a database that has an existing set of realms.
- Use Case: Testing the Addition of New Objects in a Realm
  In this use case, you add new objects to an existing realm and then test it using simulation mode without removing the current realm protections.
- Use Case: Testing the Removal of Objects from a Realm
  In this use case, you test the removal of objects to an existing realm.
- Use Case: Testing the Addition of an Authorized User to a Realm
  In this use case, you loosen security controls by adding more users. You do not need to simulate anything if you are simply adding more authorized users.
- Use Case: Testing the Removal of an Authorized User from a Realm
  In this use case, you want to drop an authorized user and use simulation mode to check if the user still needs to access the realm.
- Use Case: Testing New Factors with Realms
  In this use case, you want to test changes to factors.

- Use Case: Testing Changes to an Existing Command Rule
  In this use case, you test changes to an existing command rule while keeping the original command rule enabled.

10.3.1 Considerations When Logging Realms in Simulation Mode

There are several use cases to consider if you want to use realms in simulation mode.

- Testing an application with all new Database Vault controls: all realms are in simulation mode
- Adding a realm to existing working Database Vault controls: only a subset of realms are in simulation mode
- Adding new objects to an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Dropping one or more existing objects from an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Adding new authorized users to an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Dropping one or more existing authorized users from an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Adding or changing factors in an existing enabled realm and then testing the difference with simulation mode and not disabling existing controls
- Testing changes to a command rule in production while keeping the original command rule enabled

When a user runs a SQL statement and it fails, it may fail for realms that are enabled, fail for realms that are simulated, or it could fail for both of these reasons. There could be mandatory realms, regular realms, or both. These conditions determine the data that is captured in the simulation log.

After you create the use cases that are described in the next sections, regular realms are completely overpowered by mandatory realms when an object has both types of realms protecting it. In all cases where mandatory and regular realms protect the same object, regular realms can be ignored with regard to simulation logs. Only mandatory realm failures are captured in the simulation logs. The only time regular realm failures are entered into the simulation log is when all realms for an object are regular realms. And then, the following must happen for regular realms to be written to the simulation log:

- All regular realms in simulation mode fails and
- All regular realms that are enabled also fail

If at least one enabled or simulation regular realm succeeds, then no simulation regular realms are logged.

10.3.2 Use Case: All New Realms in Simulation Mode

In this use case, all realms are either mandatory or regular and all user-created realms are in simulation mode.

Examples are as follows:

- Mandatory realms only, which are all in simulation mode
  - The user is authorized to run the SQL statement in all mandatory realms. Nothing is captured in the simulation log table.
  - The user fails one or more mandatory realm checks. All realm check failures are logged to the simulation log. Mandatory realm checks where the user's SQL statement succeeded is not logged.

    In this example, there are three mandatory realms. The user SQL statement succeeds with one realm and fails with the other two. Only the two failed realm checks are recorded in the simulation log.
- Regular realms only, which are all in simulation mode
  - The user is authorized to run the SQL statement in at least one regular realm. The user should have access to the data so nothing is logged to the simulation log.
  - The user is not authorized to run the SQL statement in all regular realms. The simulation log captures all the failed realm authorization failures. This enables the user to select which realm to which the user should be authorized. The SQL only needs to be authorized in one regular realm to work and not all regular realms need to be updated to authorize the SQL.
- Mix of mandatory and regular realms, which are all in simulation mode
  - In this case, you capture the key realms when a user is rejected. In the case with mandatory and regular realms, the mandatory realms are the key realms. All mandatory realms must pass the authorization check for the user to gain access. In fact, regular realms could be considered superfluous when mandatory realms are protecting an object. So in cases where there are both mandatory and regular realms protecting the same object, it is only the mandatory realms that control if the SQL statement is blocked or allowed to run. It does not matter whether the user was authorized to the regular realm or not. This example follows the rules for the first scenario, for mandatory realms in simulation mode.
  - The user is authorized to run the SQL statement in all mandatory realms. Nothing is captured in the simulation log table. Even though the user may succeed or fail in one or more regular realms, nothing about regular realm failure is captured.
  - The user fails one or more mandatory realm checks. All realm check failures are logged to the simulation log. Mandatory realm checks where the user SQL statement succeeded are not be logged.

    For example, there are three mandatory realms. The user SQL statement succeeds with one realm and fails with the other two. Only the two failed realm checks are recorded in the simulation log.

    No regular realms need to be captured, because only the mandatory realms need to be captured in the simulation log.

10.3.3 Use Case: New Realms Introduced to Existing Realms

In this use case, you add a set of new realms to a database that has an existing set of realms.

The existing realms are enabled and working. The new realms are in simulation mode. This use case applies only if both simulation and enabled realms are protecting the same object.

Examples:

- New mandatory realms in simulation mode with existing enabled mandatory realms. This use case shows some additional mandatory realms for an object: adding more security to an existing object.
  - Enabled mandatory realms and mandatory realms in simulation mode all successful with user SQL statement: in this case, the SQL executes normally and nothing is captured
  - Enabled mandatory realms (at least one) fails and mandatory realms in simulation mode all successful: SQL is blocked and nothing is written to the simulation log
  - Enabled mandatory realms (at least one) fails and mandatory realms in simulation mode has one or more failures: SQL is blocked and all failing mandatory realms in simulation mode are entered into simulation log
  - Enabled mandatory realms all successful and mandatory realms in simulation mode have at least one failure: SQL is not blocked, all failed mandatory realms in simulation mode entered into simulation log
- New regular realms in simulation mode with existing enabled regular realms: More regular realms are added to a security object, providing new ways for users to access sensitive data
  - Enabled regular realms (at least one) and regular realms in simulation mode (at least one) successful: the user SQL executes normally with nothing written to simulation log
  - Enabled regular realms (at least one) is successful, and regular realms in simulation mode all fail: user SQL executes normally, nothing is entered into the simulation log
  - Enabled regular realms all fail and regular realms in simulation mode all fail: the user SQL is blocked and all regular realms in simulation mode are entered into simulation log. The user must evaluate which regular realm to authorize to if needed. The current implementation blocks the SQL and does not add the regular realms in simulation mode into the simulation log because the enabled regular realm would have blocked it anyway. This must change because the user may have added a new realm to authorize the SQL in this use case. There is no way to tell what happened if the new SQL should have worked, but is blocked by all regular realms in simulation mode as well (when one of the regular realms in simulation mode was designed to allow it to work). This would simulate an entry into the audit log for this situation.
  - Enabled regular realms all fail and regular realms in simulation mode (at least one) successful: the user SQL is blocked and nothing is written to the simulation log.
- New regular realms with existing enabled mandatory realms: You do not need to do anything in this situation. The enabled mandatory realms will continue to control the objects and the new regular realms in simulation mode will have no impact if they are enabled or not. No simulation logs should be generated in this case.
- New mandatory realms in simulation mode with existing enabled regular realm: While the enabled regular realm controls the objects for now, when the new mandatory realms in simulation mode are enabled, then they will have full control over the objects with no control by the older enabled regular realms. So, simulation logs will be created for all mandatory realms. This is the same as the scenario for new mandatory realms with existing enabled mandatory realms.

- New regular realms in simulation mode with existing enabled mandatory and regular realms: The enabled mandatory realms will be the deciding realms whether the new regular realms in simulation mode are added to the existing enabled regular realms in the system. This is the same as the scenario as a mix of mandatory and regular realms, all in simulation mode. Nothing is written to the simulation logs.
- New mandatory realms in simulation mode with enabled mandatory and regular realms: The enabled regular realms can be ignored. This is the same as the scenario for new mandatory realms with existing enabled mandatory realms.
- Mix of new mandatory and regular realms in simulation mode with existing enabled mandatory and regular realms: Ignore all enabled and mandatory regular realms. This is simply adding more mandatory realms to an existing object. This is the same scenario as new mandatory realms with existing enabled mandatory realms.

10.3.4 Use Case: Testing the Addition of New Objects in a Realm

In this use case, you add new objects to an existing realm and then test it using simulation mode without removing the current realm protections.

Oracle recommends that you create a duplicate realm in simulation mode for the new objects with the same authorized users and rule sets. This way, the existing realm can continue to provide protection to the existing objects while testing the new object.

10.3.5 Use Case: Testing the Removal of Objects from a Realm

In this use case, you test the removal of objects to an existing realm.

Because you are removing security controls for an existing object, there is no need to use simulation mode. Simply remove the object from the realm.

10.3.6 Use Case: Testing the Addition of an Authorized User to a Realm

In this use case, you loosen security controls by adding more users. You do not need to simulate anything if you are simply adding more authorized users.

If you are adding new functionality that is accessing data in a realm, but are not sure which new database users to authorize to the realm, then simply run the new functionality as a test (which will be blocked if not already authorized). Review the Database Vault audit logs to see the user name that attempted to access the realm data and add any new database users that are now authorized.

10.3.7 Use Case: Testing the Removal of an Authorized User from a Realm

In this use case, you want to drop an authorized user and use simulation mode to check if the user still needs to access the realm.

You may not be sure about dropping this user because you must check if the authorized user is accessing the realm for authorized activities.

If the data is only protected by a regular realm, then you can clone the realm with authorized users as the only difference. Remove the user to be dropped from the original realm and then add this user to the cloned realm. Then the cloned realm's audit setting is changed to capture `audit on success`. This enables the dropped user to be visible in the audit records if they accessed the realm over a period of time. Audit policies can also be used in this case. For data that is protected by a mandatory realm, the best solution is to create an audit policy.

10.3.8 Use Case: Testing New Factors with Realms

In this use case, you want to test changes to factors.

There are two scenarios where the factors can change:

- Changes to an application or the infrastructure that force a change to the factors

  In this case, you do not need to keep the original factors in place. However, objects and authorized users should be able to remain enabled during the testing of the new factors. With an enabled realm, you can remove the factors from the authorized users. At the same time, create a mandatory realm for the same protected objects in simulation mode with no authorized users. The regular realm will protect the objects from unauthorized users while the simulation realm will capture all access along with the factor information. The simulation log can then be mined for each user to come up with the new factors which can then be added to the mandatory realm in simulation mode to make sure it's clean before being migrated to the original regular realm.

- No changes to the application or the infrastructure but changes such as new factors being added or factors being removed take place

  When factors are being added, you must clone a second simulation realm from the original, but with the new factors added in. If the simulation logs shows that the usage is clean, then you can safely introduce the new factors into the original realm.

  Dropping factors lowers the security profile, so you can simply drop the factor from the rule set. No testing needs to be done.

10.3.9 Use Case: Testing Changes to an Existing Command Rule

In this use case, you test changes to an existing command rule while keeping the original command rule enabled.

Command rules may need to be updated and ideally tested before the changes are enabled in production. For a new command rule that will be added to a set of already existing command rules, put the new command rule into simulation mode when you create it. The other pre-existing command rules are already enabled and offer protection.

If you want to modify an existing command rule, there is no way to maintain the existing protection and test the new modification. Oracle recommends that you create an audit policy to capture what the original command rule was doing and then set an alert for it. The audit will not prevent the SQL as a command rule would do, but at least you can be alerted about the action. Then you can put the new updated command rule into simulation mode and test it.

10.4 Tutorial: Tracking Violations to a Realm Using Simulation Mode

This tutorial shows how to create a realm that uses simulation mode and then test violations to the realm.

- About This Tutorial
  In this tutorial, you will create a realm around the HR.EMPLOYEES table and test violations against it.
- Step 1: Create Users for This Tutorial
  You must create three users for this tutorial.
- Step 2: Create a Realm and an Oracle Database Vault Policy
  Next, you create a realm around the HR.EMPLOYEES table, and then add this realm to an Oracle Database Vault policy.
- Step 3: Test the Realm and Policy
  User tjones_dba will commit a violation on the realm to test the realm and policy.
- Step 4: Query the DBA_DV_SIMULATION_LOG View for Violations
  Now you can check the simulation mode log for the violations that user tjones_dba committed.
- Step 5: Enable and Re-test the Realm
  Now that you have captured the violations, user psmith can update the HR.EMPLOYEES_pol policy.
- Step 6: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

10.4.1 About This Tutorial

In this tutorial, you will create a realm around the HR.EMPLOYEES table and test violations against it.

The HR.EMPLOYEES table contains confidential data such employee salaries. To test the realm, an administrator, tjones_dba, will look up and modify the salary of another employee, smavris. The Database Vault administrator, sec_admin_owen, will use simulation mode to track the violations to the HR.EMPLOYEES table. To accomplish this, user sec_admin_owen will create a Database Vault policy, which a delegated administrator, user psmith, will own. User psmith will then be able to make limited changes to the policy without needing the DV_OWNER or DV_ADMIN role.

10.4.2 Step 1: Create Users for This Tutorial

You must create three users for this tutorial.

The users are: psmith, who is the Database Vault policy owner; tjones_dba, who commits violations on the HR.EMPLOYEES table; and smavris, whose salary is the recipient of tjones_dba's violations.

1. Log in to a PDB as a user who has been granted the DV_ACCTMGR role.
2. Create the following users and grant them the CREATE SESSION privilege.

   ```
   GRANT CREATE SESSION TO psmith IDENTIFIED BY password;
   GRANT CREATE SESSION TO tjones_dba IDENTIFIED BY password;
   GRANT CREATE SESSION TO smavris IDENTIFIED BY password;
   ```

   Replace *password* with a password that meets the password complexity requirements of the user's profile.
3. Connect as a user who has been granted the DV_OWNER role.

For example:

```
CONNECT sec_admin_owen@pdb_name
Enter password: password
```

4. Grant user `psmith` the `DV_POLICY_OWNER` role, which enables `psmith` to manage Database Vault policies.

```
GRANT DV_POLICY_OWNER TO psmith;
```

5. Connect as user `SYS` with the `SYSDBA` administrative privilege.
6. Grant the `DBA` role to user `tjones_dba`

```
GRANT DBA TO tjones_dba;
```

7. Connect as the `HR` schema owner.

```
CONNECT HR@pdb_name
Enter password: password
```

8. Grant the `SELECT` privilege on the `HR.EMPLOYEES` table to user `smavris`

```
GRANT SELECT ON HR.EMPLOYEES TO smavris;
```

At this stage, the users have all been created and granted the appropriate privileges.

Related Topics

- *Oracle Database Security Guide*

10.4.3 Step 2: Create a Realm and an Oracle Database Vault Policy

Next, you create a realm around the `HR.EMPLOYEES` table, and then add this realm to an Oracle Database Vault policy.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` role.
2. Create the realm around `HR.EMPLOYEES` table as follows.

   These procedures create the `HR.EMPLOYEES_realm` realm, add the `HR.EMPLOYEES` table to this realm, authenticate `HR` as an owner, authenticate user `psmith` as an participant, and set the realm in simulation mode.

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'HR.EMPLOYEES_realm',
  description   => 'Realm to protect HR.EMPLOYEES',
  enabled       => DBMS_MACUTL.G_SIMULATION,
  audit_options => DBMS_MACUTL.G_REALM_AUDIT_OFF,
  realm_type    => 0);
END;
/

BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'HR.EMPLOYEES_realm',
  object_owner => 'HR',
  object_name  => 'EMPLOYEES',
  object_type  => 'TABLE');
END;
/
```

3. Create the `HR.EMPLOYEES_pol` Database Vault policy and set it to be in simulation mode.

   These procedures create the `HR.EMPLOYEES_pol` policy, add the realm that was just created to the policy, and then add user `psmith` as the owner of the policy.

```
BEGIN
 DBMS_MACADM.CREATE_POLICY(
  policy_name  => 'HR.EMPLOYEES_pol',
  description  => 'Policy to protect HR.EMPLOYEES',
  policy_state => DBMS_MACADM.G_SIMULATION);
END;
/

BEGIN
 DBMS_MACADM.ADD_REALM_TO_POLICY(
  policy_name  => 'HR.EMPLOYEES_pol',
  realm_name   => 'HR.EMPLOYEES_realm');
END;
/

BEGIN
 DBMS_MACADM.ADD_OWNER_TO_POLICY(
  policy_name  => 'HR.EMPLOYEES_pol',
  owner_name   => 'PSMITH');
END;
/
```

At this point, the realm and policy are ready to be tested.

10.4.4 Step 3: Test the Realm and Policy

User `tjones_dba` will commit a violation on the realm to test the realm and policy.

1. Connect to the PDB as user `tjones_dba`.
2. Query the `HR.EMPLOYEES` table for the salary of `smavris`.

```
SELECT SALARY FROM HR.EMPLOYEES WHERE EMAIL = 'SMAVRIS';
```

   Output similar to the following should appear:

```
    SALARY
----------
      6500
```

3. Cut `smavris`'s salary in half.

```
UPDATE HR.EMPLOYEES
SET SALARY = SALARY / 2
WHERE EMAIL = 'SMAVRIS';

1 row updated.
```

4. Connect as user `smavris`.
5. Query the salary of `smavris`.

```
SELECT SALARY FROM HR.EMPLOYEES WHERE EMAIL = 'SMAVRIS';
```

   Output similar to the following should appear:

```
    SALARY
----------
      3250
```

At this point, `tjones_dba`'s violations have been recorded in the `DBA_DV_SIMULATION_LOG` data dictionary view.

10.4.5 Step 4: Query the DBA_DV_SIMULATION_LOG View for Violations

Now you can check the simulation mode log for the violations that user `tjones_dba` committed.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` role.
2. Query the `DBA_DV_SIMULATION_LOG` data dictionary view.

```
SELECT USERNAME, COMMAND, SQLTEXT, VIOLATION_TYPE
FROM DBA_DV_SIMULATION_LOG
WHERE REALM_NAME = 'HR.EMPLOYEES_realm';
```

Output similar to the following should appear:

```
USERNAME
--------------------------------------------------------------------------------
---
COMMAND
--------------------------------------------------------------------------------
---
SQLTEXT
--------------------------------------------------------------------------------
---
VIOLATION_TYPE
--------------------------------------------------------------------------------
---
TJONES_DBA
UPDATE
UPDATE HR.EMPLOYEES SET SALARY = SALARY / 2 WHERE EMAIL = 'SMAVRIS'
Realm Violation

USERNAME
--------------------------------------------------------------------------------
---
COMMAND
--------------------------------------------------------------------------------
---
SQLTEXT
--------------------------------------------------------------------------------
---
VIOLATION_TYPE
--------------------------------------------------------------------------------
---
TJONES_DBA
SELECT
SELECT SALARY FROM HR.EMPLOYEES WHERE EMAIL = 'SMAVRIS'
Realm Violation
```

The output indicates that user `tjones_dba` has committed two offences: first, `tjones_dba` looked at another employee's salary, and not only that, `tjones_dba` cut it

in half. The violation type is a realm violation. The query by `smavris` was not captured because she legitimately can look at `smavris`'d salary.

10.4.6 Step 5: Enable and Re-test the Realm

Now that you have captured the violations, user `psmith` can update the `HR.EMPLOYEES_pol` policy.

This is so that the `HR.EMPLOYEES_realm` realm can be enabled. Then you can test the violations again.

1. Connect to the PDB as user `psmith`.
2. Update the policy so that it is enabled.

```
BEGIN
 DBMS_MACADM.UPDATE_POLICY_STATE(
  policy_name   => 'HR.EMPLOYEES_pol',
  policy_state  => 1);
END;
/
```

3. Connect as user `tjones_dba`.

```
CONNECT tjones_dba@pdb_name
```

4. Try lowering `smavris`'s salary to new depths.

```
UPDATE HR.EMPLOYEES
SET SALARY = SALARY / 2
WHERE EMAIL = 'SMAVRIS';
```

Output similar to the following should appear:

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

The policy, now enabled, enables the realm to protect the `HR.EMPLOYEES` table. `smavris`'s salary can shrink no more.

10.4.7 Step 6: Remove the Components for This Tutorial

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

1. Connect to the PDB as a user who has been granted the `DV_OWNER` role.
2. Remove the `HR.EMPLOYEES_pol` Database Vault policy.

```
EXEC DBMS_MACADM.DROP_POLICY('HR.EMPLOYEES_pol');
```

You first must remove the policy before you can drop its contents.

3. Remove the `HR.EMPLOYEES_realm` realm.

```
EXEC DBMS_MACADM.DELETE_REALM('HR.EMPLOYEES_realm');
```

4. Remove the simulation mode log data that was accumulated.

Because the simulation mode log only captured information about user `tjones_dba`, you can remove only the rows that relate to this user.

```
DELETE FROM DVSYS.SIMULATION_LOG$ WHERE USERNAME = 'TJONES_DBA';
```

5. Connect as user `HR`.

```
CONNECT HR@pdb_name
Enter password: password
```

6. Revert `smavris`'s salary back to its pre-violated state.

```
UPDATE HR.EMPLOYEES
SET SALARY = 6500
WHERE EMAIL = 'SMAVRIS';
```

7. Connect as a user who has been granted the `DV_ACCTMGR` role.

   For example:

```
CONNECT accts_admin_ace@pdb_name
Enter password: password
```

8. Remove the users `psmith`, `smavris`, and `tjones_dba`.

```
DROP USER psmith;
DROP USER smavris;
DROP USER tjones_dba;
```

11

Integrating Oracle Database Vault with Other Oracle Products

You can integrate Oracle Database Vault with other Oracle products, such as Oracle Data Guard.

- Integrating Oracle Database Vault with Transparent Data Encryption
  Transparent Data Encryption complements Oracle Database Vault in that it provides data protection when the data leaves the secure perimeter of the database.
- Attaching Factors to an Oracle Virtual Private Database
  You can attach Oracle Database Vault factors to an Oracle Virtual Private Database.
- Integrating Oracle Database Vault with Oracle Label Security
  You can integrate Oracle Database Vault with Oracle Label Security, and check the integration with reports and data dictionary views.
- Integrating Oracle Database Vault with Oracle Data Guard
  An Oracle Database Vault-Oracle Data Guard integration requires first, the primary database configuration, then the standby database configration.
- Configuring Oracle Internet Directory Using Oracle Database Configuration Asssitant
  You can use Oracle Internet Directory in an Oracle Database Vault-enabled database.
- Integrating Oracle Database Vault with Enterprise User Security
  You can integrate Oracle Database Vault with Oracle Enterprise User Security.
- Integrating Oracle Database Vault with Oracle APEX
  You can integrate Oracle Database Vault with Oracle APEX.

11.1 Integrating Oracle Database Vault with Transparent Data Encryption

Transparent Data Encryption complements Oracle Database Vault in that it provides data protection when the data leaves the secure perimeter of the database.

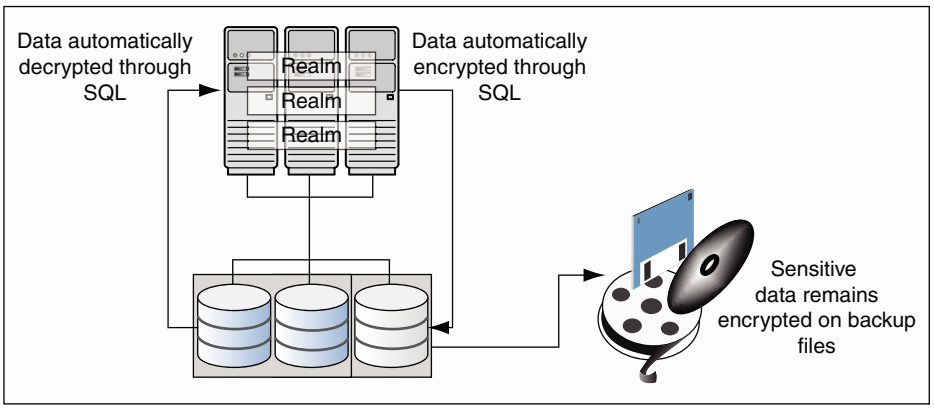
With Transparent Data Encryption, a database administrator or database security administrator can simply encrypt columns with sensitive content in application tables, or encrypt entire application tablespaces, without any modification to the application.

If a user passes the authentication and authorization checks, Transparent Data Encryption automatically encrypts and decrypts information for the user. This way, you can implement encryption without having to change your applications.

Once you have granted the Transparent Data Encryption user the appropriate privileges, then Transparent Data Encryption can be managed as usual and be used complimentary to Database Vault.

Figure 11-1 shows how Oracle Database Vault realms handle encrypted data.

Figure 11-1 Encrypted Data and Oracle Database Vault



Related Topics

- *Oracle Database Advanced Security Guide*

11.2 Attaching Factors to an Oracle Virtual Private Database

You can attach Oracle Database Vault factors to an Oracle Virtual Private Database.

1. Define a Virtual Private Database policy predicate that is a PL/SQL function or expression.
2. For each function or expression, use the `DVF.F$` PL/SQL function that is created for each factor.

Related Topics

- *Oracle Database Security Guide*

11.3 Integrating Oracle Database Vault with Oracle Label Security

You can integrate Oracle Database Vault with Oracle Label Security, and check the integration with reports and data dictionary views.

- How Oracle Database Vault Is Integrated with Oracle Label Security
  An Oracle Database Vault-Oracle Label Security integration enables you to assign an OLS label to a Database Vault factor identity.
- Requirements for Using Oracle Database Vault with Oracle Label Security
  You must fulfill specific requirements in place before you use Oracle Database Vault with Oracle Label Security.
- Using Oracle Database Vault Factors with Oracle Label Security Policies
  To enhance security, you can integrate Oracle Database Vault factors with Oracle Label Security policies.
- Tutorial: Integrating Oracle Database Vault with Oracle Label Security
  An Oracle Database Vault-Oracle Label Security integration can grant different levels of access to two administrative users who have the same privileges.

- Related Reports and Data Dictionary Views
  Oracle Database Vault provides reports and data dictionary views that list information about the Oracle Database Vault-Oracle Label Security integration.

11.3.1 How Oracle Database Vault Is Integrated with Oracle Label Security

An Oracle Database Vault-Oracle Label Security integration enables you to assign an OLS label to a Database Vault factor identity.

In Oracle Label Security, you can restrict access to records in database tables or PL/SQL programs. For example, Mary may be able to see data protected by the `HIGHLY SENSITIVE` label, an Oracle Label Security label on the `EMPLOYEE` table that includes records that should have access limited to certain managers. Another label can be `PUBLIC`, which allows more open access to this data.

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

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

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

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

This means that when Mary is working at home using their VPN connection, she have 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.

The Label Security Integration Audit Report enables you to monitor the integration with Oracle Label Security. You can create audit policies to capture this information. Be aware that as of Oracle Database release 23c, traditional auditing is desupported.

Related Topics

- Tutorial: Integrating Oracle Database Vault with Oracle Label Security
  An Oracle Database Vault-Oracle Label Security integration can grant different levels of access to two administrative users who have the same privileges.
- Oracle Database Vault Oracle Label Security APIs
  You can use the `DBMS_MACADM` PL/SQL package to manage Oracle Label Security labels and policies in Oracle Database Vault.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.
- *Oracle Label Security Administrator's Guide*

11.3.2 Requirements for Using Oracle Database Vault with Oracle Label Security

You must fulfill specific requirements in place before you use Oracle Database Vault with Oracle Label Security.

- Oracle Label Security is licensed separately. Ensure that you have purchased a license to use it.
- Before you install Oracle Database Vault, you must have already installed Oracle Label Security.
- The installation process for Oracle Label Security creates the `LBACSYS` user account. As a user who has been granted the `DV_ACCTMGR` role, unlock this account and grant it a new password. For example:

```
sqlplus accts_admin_ace@pdb_name
Enter password: password

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

- If you plan to use the `LBACSYS` user account in Oracle Enterprise Manager, then log in to Enterprise Manager as user `SYS` with the `SYSDBA` administrative privilege, and grant this user the `SELECT ANY DICTIONARY` and `SELECT_CATALOG_ROLE` system privileges.
- Ensure that you have the appropriate Oracle Label Security policies defined.
- If you plan to integrate an Oracle Label Security policy with a Database Vault policy, then ensure that the policy name for Oracle Label Security is less than 24 characters. You can check the names of Oracle Label Security policies by querying the `POLICY_NAME` column of the `ALL_SA_POLICIES` data dictionary view.

11.3.3 Using Oracle Database Vault Factors with Oracle Label Security Policies

To enhance security, you can integrate Oracle Database Vault factors with Oracle Label Security policies.

- About Using Oracle Database Vault Factors with Oracle Label Security Policies
  And Oracle Database Vault-Oracle Label Security integration enables you to control the maximum security clearance for a database session.
- Configuring Factors to Work with an Oracle Label Security Policy
  You can define factors that contribute to the maximum allowable data label of an Oracle Label Security policy.

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

And Oracle Database Vault-Oracle Label Security integration enables you to control the maximum security clearance for a database session.

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

In brief, a label acts as an identifier for the access privileges of a database table row. A policy is a name associated with the labels, rules, and authorizations that govern access to table rows.

Related Topics

- *Oracle Label Security Administrator's Guide*

11.3.3.2 Configuring Factors to Work with an Oracle Label Security Policy

You can define factors that contribute to the maximum allowable data label of an Oracle Label Security policy.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
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.

   For example, to make `LBACSYS` the owner of a realm called `HR Realm`:

   ```
   BEGIN
    DBMS_MACADM.ADD_AUTH_TO_REALM(
     realm_name    => 'HR Realm',
     grantee       => 'LBACSYS',
     auth_options => DBMS_MACUTL.G_REALM_AUTH_OWNER);
   END;
   /
   ```

3. Authorize the schema owner (on which the label security policy has been applied) as either a realm participant or a realm owner.

   For example:

   ```
   BEGIN
    DBMS_MACADM.ADD_AUTH_TO_REALM(
     realm_name    => 'HR Realm',
     grantee       => 'HR',
     auth_options => DBMS_MACUTL.G_REALM_AUTH_OWNER);
   END;
   /
   ```

4. Configure a label security policy for the realm.

   Optionally set the label-merging algorithm for cases when Oracle Label Security has merged two labels. In most cases, you may want to configure the label security policy to use `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.

   - To create a new label security policy, run the `DBMS_MACADM.CREATE_MAC_POLICY` procedure. For example:

     ```
     BEGIN
      DBMS_MACADM.CREATE_MAC_POLICY(
       policy_name  => 'Access Locations',
       algorithm    => 'LII');
     END;
     /
     ```

- To modify an existing label security policy, run the `DBMS_MACADM.UPDATE_MAC_POLICY` procedure.

5. Run the `DBMS_MACADM.ADD_POLICY_FACTOR` factor to associate a factor with the label security policy.

   For example:

```
BEGIN
 DBMS_MACADM.ADD_POLICY_FACTOR(
  policy_name  => 'Access Locations',
  factor_name  => 'Sector2_DB');
END;
/
```

6. Run the `DBMS_MACADM.CREATE_IDENTITY` procedure to create a factor identity.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_IDENTITY(
  factor_name  => 'Sector2_DB',
  value        => 'intranet',
  trust_level  => 5);
END;
/
```

7. Label the factor identities using the labels for the policy.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_POLICY_LABEL(
  identity_factor_name   => 'Sector2_DB',
  identity_factor_value  => 'intranet',
  policy_name            => 'Access Locations',
  label                  => 'sensitive');
END;
/
```

> **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.

Related Topics

- 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. You can authenticate both common and local realms.
- Oracle Database Vault Oracle Label Security APIs
  You can use the `DBMS_MACADM` PL/SQL package to manage Oracle Label Security labels and policies in Oracle Database Vault.
- Oracle Database Vault Factor APIs
  The `DBMS_MACADM` PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and `DVF` has functions to manage factors.

11.3.4 Tutorial: Integrating Oracle Database Vault with Oracle Label Security

An Oracle Database Vault-Oracle Label Security integration can grant different levels of access to two administrative users who have the same privileges.

- About This Tutorial
  You can use Oracle Database Vault factors with Oracle Label Security and Oracle Virtual Private Database (VPD) to restrict sensitive data access.
- Step 1: Create Users for This Tutorial
  You must create two administrative users for this tutorial.
- Step 2: Create the Oracle Label Security Policy
  Next, you can create the Oracle Label Security policy and grant users the appropriate privileges for it.
- Step 3: Create Oracle Database Vault Rules to Control the OLS Authorization
  After you create the Oracle Label Security policy, you can create Database Vault rules to work with it.
- Step 4: Update the ALTER SYSTEM Command Rule to Use the Rule Set
  Before the rule set can be used, you must update the ALTER SYSTEM command rule, which is a default command rule.
- Step 5: Test the Authorizations
  With all the components in place, you are ready to test the authorization.
- Step 6: Remove the Components for This Tutorial
  You can remove the components that you created for this tutorial if you no longer need them.

11.3.4.1 About This Tutorial

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

You can restrict this data so that it is only exposed to a database session when the correct combination of factors exists, defined by the security administrator, for any given database session.

11.3.4.2 Step 1: Create Users for This Tutorial

You must create two administrative users for this tutorial.

1. Log in to a PDB as a user who has been granted the `DV_ACCTMGR` role.
2. Create the following local users:

   ```
   GRANT CREATE SESSION TO mdale IDENTIFIED BY password CONTAINER = CURRENT;
   GRANT CREATE SESSION TO jsmith IDENTIFIED BY password CONTAINER = CURRENT;
   ```

   Replace *password* with a password that meets the password complexity requirements of the user's profile.
3. Connect as a user who can grant system privileges and who has been granted the owner authorization for the Oracle System Privilege and Role Management realm, and then grant administrative privileges to users `mdale` and `jsmith`.

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

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

Related Topics

- *Oracle Database Security Guide*

11.3.4.3 Step 2: Create the Oracle Label Security Policy

Next, you can create the Oracle Label Security policy and grant users the appropriate privileges for it.

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

   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:

   ```
   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.

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

After you create the Oracle Label Security policy, you can create Database Vault rules to work with it.

1. Connect to the PDB as the Database Vault Owner.
2. Create the following rule set:

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

3. Create a rule for the PII Rule Set.

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

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

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

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

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

Before the rule set can be used, you must update the ALTER SYSTEM command rule, which is a default command rule.

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

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

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

   In a default installation, the ALTER SYSTEM command rule uses the Allow Fine Grained Control of System Parameters rule set, and is enabled.

3. Update the ALTER SYSTEM command rule to be associated with the PII Rule Set.

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

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

11.3.4.6 Step 5: Test the Authorizations

With all the components in place, you are ready to test the authorization.

1. In SQL*Plus, log in to the PDB as user `mdale`.
2. Check the current setting for the `CPU_COUNT` initialization parameter.

```
SHOW PARAMETER PARAMETER CPU_COUNT

NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------------------
cpu_count                            string      2
```

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

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

```
ALTER SYSTEM SET CPU_COUNT = 4;
System altered.
```

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

4. Set the `CPU_COUNT` parameter back to its original value.

   For example:

```
ALTER SYSTEM SET CPU_COUNT = 2;
```

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

```
ALTER SYSTEM SET CPU_COUNT = 14;
```

The following output should appear:

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

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

11.3.4.7 Step 6: Remove the Components for This Tutorial

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

1. Connect to the PDB as the Oracle Label Security administrator and remove the label policy and its components.

```
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.

```
EXEC DBMS_MACADM.UPDATE_COMMAND_RULE('ALTER SYSTEM', 'Allow System
Parameters','%', '%', 'Y');
EXEC DBMS_MACADM.DELETE_RULE_FROM_RULE_SET('PII Rule Set', 'Check OLS
Factor');
EXEC DBMS_MACADM.DELETE_RULE('Check OLS Factor');
EXEC DBMS_MACADM.DELETE_RULE_SET('PII Rule Set');
COMMIT;
```

3. Connect as the Database Vault Account Manager and remove users `mdale` and `jsmith`.

```
DROP USER mdale;
DROP USER jsmith;
```

11.3.5 Related Reports and Data Dictionary Views

Oracle Database Vault provides reports and data dictionary views that list information about the Oracle Database Vault-Oracle Label Security integration.

Table 11-1 lists the Oracle Database Vault reports.

Table 11-1 Reports Related to Database Vault and Oracle Label Security Integration

Report	Description
Factor Configuration Issues Report	Lists factors in which the Oracle Label Security policy does not exist.
Identity Configuration Issues Report	Lists invalid label identities (the Oracle Label Security label for this identity has been removed and no longer exists).

Table 11-1 (Cont.) Reports Related to Database Vault and Oracle Label Security Integration

Report	Description
Security Policy Exemption Report	Lists accounts and roles that have the `EXEMPT ACCESS POLICY` system privilege granted to them. Accounts that have this privilege can bypass all Virtual Private Database policy filters and any Oracle Label Security policies that use Oracle Virtual Private Database indirectly.

Table 11-2 lists data dictionary views that provide information about existing Oracle Label Security policies used with Oracle Database Vault.

Table 11-2 Data Dictionary Views Used for Oracle Label Security

Data Dictionary View	Description
`DBA_DV_MAC_POLICY`	Lists the Oracle Label Security policies defined
`DBA_DV_MAC_POLICY`	Lists the factors that are associated with Oracle Label Security policies
`DBA_DV_POLICY_LABEL`	Lists the Oracle Label Security label for each factor identifier in the `DBA_DV_IDENTITY` view for each policy

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

11.4 Integrating Oracle Database Vault with Oracle Data Guard

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

- Step 1: Configure the Primary Database
  You must run the `DGMGRL` utility, configure and enable Oracle Database Vault, and then run the `ALTER SYSTEM` statement, to configure the primary database.
- Step 2: Configure the Standby Database
  You can perform the standby database configuration within the database to be used for the standby database.
- How Auditing Works After an Oracle Database Vault-Oracle Active Data Guard Integration
  After you have integrated Oracle Database Vault with Oracle Active Data Guard, how auditing is configured affects how audit records are generated.
- Disabling Oracle Database Vault in an Oracle Data Guard Environment
  If you want to disable Oracle Database Vault in an Oracle Data Guard environment, you must perform the procedures first on the primary database, and then on the standby database.

11.4.1 Step 1: Configure the Primary Database

You must run the DGMGRL utility, configure and enable Oracle Database Vault, and then run the ALTER SYSTEM statement, to configure the primary database.

1. For Linux and UNIX systems, ensure there is an /etc/oratab entry for the database on the node in which you are installing Oracle Database Vault.
2. If you are using Data Guard Broker, then from the command prompt, disable the configuration as follows:

```
dgmgrl sys
Enter password: password

DGMGRL> disable configuration;
```

3. Configure and enable Oracle Database Vault on the primary server.

   By default, Oracle Database Vault is installed as part of Oracle Database. You can check the registration status by querying the DBA_DV_STATUS data dictionary view.
4. Log in to the PDB as user SYS with the SYSDBA administrative privilege.
5. Run the following ALTER SYSTEM statements:

```
ALTER SYSTEM SET OS_ROLES=FALSE SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_LOGIN_PASSWORDFILE='EXCLUSIVE' SCOPE=SPFILE;
ALTER SYSTEM SET SQL92_SECURITY=TRUE SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_OS_ROLES=FALSE SCOPE=SPFILE;
```

6. Run the ALTER SYSTEM statement on each database instance to set the parameters as shown in the preceding step.
7. As user SYS with the SYSDBA administrative privilege, close and then reopen each PDB.

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

Related Topics

- Getting Started with Oracle Database Vault
  Before you can start using Oracle Database Vault, you must configure and enable it with the Oracle database.

11.4.2 Step 2: Configure the Standby Database

You can perform the standby database configuration within the database to be used for the standby database.

1. Log into the database instance as user SYS with the SYSDBA administrative privilege.
2. Mount a standby database instance.

```
ALTER DATABASE MOUNT STANDBY DATABASE;
```

3. Run the following ALTER SYSTEM statements:

```
ALTER SYSTEM SET OS_ROLES=FALSE SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_LOGIN_PASSWORDFILE='EXCLUSIVE' SCOPE=SPFILE;
```

```
ALTER SYSTEM SET SQL92_SECURITY=TRUE SCOPE=SPFILE;
ALTER SYSTEM SET REMOTE_OS_ROLES=FALSE SCOPE=SPFILE;
```

4. Close and then reopen the PDB.

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

5. Mount the next standby instance.
6. Restart the managed recovery as follows:

   ```
   ALTER DATABASE RECOVER MANAGED STANDBY DATABASE;
   ```

7. If you are using Data Guard Broker, then from the command line, re-enable the configuration.

   ```
   dgmgrl sys
   Enter password: password

   DGMGRL> enable configuration;
   ```

   This command applies the changes to the physical standby database made by the Oracle Database Vault installation on the primary database.
8. Repeat the physical standby installation process on each physical standby database. For example, if there are three physical standby databases, then run these procedures or each standby database.

11.4.3 How Auditing Works After an Oracle Database Vault-Oracle Active Data Guard Integration

After you have integrated Oracle Database Vault with Oracle Active Data Guard, how auditing is configured affects how audit records are generated.

If you want to use the Active Data Guard physical standby database for read-only queries, then you must use unified auditing. Oracle Database Vault cannot write to the traditional Database Vault audit table (`DVSYS.AUDIT_TRAILS$`). Unified auditing will ensure that the Database Vault audit data is written into the operating system log files in an Oracle Active Data Guard physical standby database. You can move the data in these log files to the unified audit trail. Remember that to audit Database Vault activities, you must create unified audit policies, because traditional auditing is no longer supported.

Related Topics

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

11.4.4 Disabling Oracle Database Vault in an Oracle Data Guard Environment

If you want to disable Oracle Database Vault in an Oracle Data Guard environment, you must perform the procedures first on the primary database, and then on the standby database.

Perform the disablement of Oracle Database Vault on the primary and standby databases in the following order:

1. Disable Oracle Database Vault on the primary database.

2. Disable Oracle Database Vault on the secondary database.
3. Restart the primary database.
4. Restart each standby database.

Related Topics

- Step 1: Disable Oracle Database Vault
  Be aware that after you disable Oracle Database Vault, Oracle Label Security, which is required to run Database Vault, is still enabled.

11.5 Configuring Oracle Internet Directory Using Oracle Database Configuration Asssitant

You can use Oracle Internet Directory in an Oracle Database Vault-enabled database.

However, if you want to configure Oracle Internet Directory (OID) using Oracle Database Configuration Assistant (DBCA), then you must first disable Oracle Database Vault.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

11.6 Integrating Oracle Database Vault with Enterprise User Security

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

- About Integrating Oracle Database Vault with Enterprise User Security
  Enterprise User Security centrally manages database users and authorizations in one place.
- Configuring an Enterprise User Authorization
  To configure an Enterprise User authorization, you must create an Oracle Database Vault rule set to control the user access.
- Configuring Oracle Database Vault Accounts as Enterprise User Accounts
  You can configure existing Oracle Database Vault user accounts as enterprise user accounts in a PDB.

11.6.1 About Integrating Oracle Database Vault with Enterprise User Security

Enterprise User Security centrally manages database users and authorizations in one place.

It is combined with Oracle Identity Management and is available in Oracle Database Enterprise Edition.

In general, to integrate Oracle Database Vault with Oracle Enterprise User Security, you configure the appropriate realms to protect the data that you want to protect in the database.

After you define the Oracle Database Vault realms as needed, you can create a rule set for the Enterprise users to allow or disallow their access.

> **Note:**
>
> Enterprise User Security (EUS) is deprecated with Oracle Database 23c. Oracle recommends that you migrate to using Centrally Managed Users (CMU). This feature enables you to directly connect with Microsoft Active Directory without an intervening directory service for enterprise user authentication and authorization to the database. If your Oracle Database is in the cloud, you can also choose to move to one of the newer integrations with a cloud identity provider.

Related Topics

- *Oracle Database Enterprise User Security Administrator's Guide*

11.6.2 Configuring an Enterprise User Authorization

To configure an Enterprise User authorization, you must create an Oracle Database Vault rule set to control the user access.

1. Connect to the PDB or the application root as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Run the `DBMS_MACADM.CREATE_RULE` procedure to create the rule that allows or disallows user access.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Control User Access',
  rule_expr  =>'SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY') =
'user_domain_name'',
  scope      => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

   In this specification:

   - `rule_name` specifies a rule name. Enter any valid name that is appropriate for your needs.
   - `rule_expr` must use the rule expression given in this example. Replace *`'user_domain_name'`* with the domain. For example:

     ```
     'SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY') = 'myserver.us.example.com''
     ```

   - `scope` must be `DBMS_MACUTL.G_SCOPE_LOCAL`.

3. Run the `DBMS_MACADM.CREATE_RULE_SET` procedure to create a rule set to be used for the rule.

   For example:

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
```

```
  rule_set_name    => 'EM User Authorization',
  description      => 'Allows or disallows user access to EM',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
  fail_message     => '',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
  handler          => ' ',
  is_static        => TRUE);
END;
/
```

4. Run the `DBMS_MACADM.ADD_RULE_TO_RULE_SET` procedure to add the rule to the rule set.

   For example:

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'EM User Authorization',
  rule_name     => 'Control User Access',
  rule_order    => 1);
END;
/
```

5. Run the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure to add the rule set to the realm authorization for the data that you want to protect.

   For example, for a realm called `HR Realm`:

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    => 'HR Realm',
  grantee       => 'pfitch',
  rule_set_name => 'EM User Authorization',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER,
  auth_scope    => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

11.6.3 Configuring Oracle Database Vault Accounts as Enterprise User Accounts

You can configure existing Oracle Database Vault user accounts as enterprise user accounts in a PDB.

1. Log in to the PDB as a user who has been granted the `CREATE ROLE` system privilege.
2. Create a global role for the `DV_OWNER` role and a global role for the `DV_ACCTMGR` role.

   For example:

```
CREATE ROLE g_dv_owner IDENTIFIED GLOBALLY;
CREATE ROLE g_dv_acctmgr IDENTIFIED GLOBALLY;
```

3. Connect as a user who has been granted the `DV_OWNER` role.
4. Grant the `DV_OWNER` role to the global `DV_OWNER` role.

```
GRANT DV_OWNER TO g_dv_owner;
```

5. Connect as a user who has been granted the DV_ACCTMGR role.
6. Grant the DV_ACCTMGR role to the global DV_ACCTMGR role.

   ```
   GRANT DV_ACCTMGR TO g_dv_acctmgr;
   ```

7. Connect as user SYS with the SYSDBA administrative privilege.
8. Temporarily grant the DV_ACCTMGR user who will import the Database Vault users into OID the CREATE TABLE privilege and the SELECT_CATALOG_ROLE role.

   ```
   GRANT CREATE TABLE, SELECT_CATALOG_ROLE TO dbv_acctmgr;
   ```

9. From the command line, run the User Migration Utility (UMU) to import the Database Vault accounts into Oracle Internet Directory (OID).

   The following example imports the Database Vault accounts sec_admin_owen and accts_admin_ace into OID. The DV_ACCTMGR user is specified for the DBADMIN setting.

   ```
   $ORACLE_HOME/rdbms/bin/umu PHASE=ONE
   DBADMIN=dbv_acctmgr:password
   ENTADMIN=cn=jane_ent_admin,dc=example,dc=com:password
   USERS= LIST
   DBLOCATION=example.com:7777:orcl
   DIRLOCATION=example.com:636
   USERSLIST=sec_admin_owen:accts_admin_ace
   MAPSCHEMA=PRIVATE
   CONTEXT=CONTEXT="c=Users, c=us"
   KREALM=EXAMPLE.COM

   $ORACLE_HOME/rdbms/bin/umu PHASE=TWO
   DBADMIN=dbv_acctmgr:password
   ENTADMIN=cn=jane_ent_admin,dc=example,dc=com:password
   DBLOCATION=example.com:7777:orcl
   DIRLOCATION=example.com:636
   ```

   By default, errors are written to the $ORACLE_HOME/network/log/umu.log file.

   Enterprise User Security (EUS) User Migration Utility (UMU) is deprecated in Oracle Database 21c. Use EUS Manager (EUSM) features instead.

10. From the Oracle Internet Directory Self Service Console (http://*hostname*:*port*/oiddas/), grant the global DV_OWNER and DV_ACCTMGR roles (for example, g_dv_owner and g_dv_acctmgr) to the enterprise user Database Vault accounts.

    See the example of creating enterprise users in *Oracle Database Enterprise User Security Administrator's Guide* for a demonstration of creating an enterprise role from a global role and then granting this role to a user.

11. From SQL*Plus, as user SYS with the SYSDBA administrative privilege, revoke the CREATE TABLE and SELECT_CATALOG_ROLE role from the DV_ACCTMGR user.

    ```
    REVOKE CREATE TABLE, SELECT_CATALOG_ROLE FROM dbv_acctmgr;
    ```

Related Topics

- *Oracle Database Enterprise User Security Administrator's Guide*

11.7 Integrating Oracle Database Vault with Oracle APEX

You can integrate Oracle Database Vault with Oracle APEX.

- About Integrating Oracle Database Vault with Oracle APEX
  Oracle APEX is Oracle's primary tool for developing Web applications with SQL and PL/SQL.
- Installing or Upgrading Oracle APEX with Oracle Database Vault Enabled
  When Oracle Database Vault is enabled, additional privileges are required to install or upgrade Oracle APEX.
- Authorizing the Oracle APEX Schema for Oracle Database Vault Activities
  You must add the Oracle APEX schema (for example, `APEX_SCHEMA`) to Oracle Database Vault realms and authorizations that are required by Oracle APEX.
- Authorizing Oracle APEX to Use Oracle Scheduler
  Oracle APEX uses Oracle Scheduler and must be authorized to continue to do so.
- Authorizing Oracle APEX to Perform DDL Tasks
  You must authorize the Oracle APEX schema to use its DDL privileges on objects that it has access to but may be subject to additional Oracle Database Vault controls
- Authorizing Oracle APEX to Perform Information Lifecycle Maintenance Tasks
  You must authorize the Oracle APEX schema to perform maintenance tasks.
- Authorizing Oracle APEX to Proxy Users for Oracle Rest Data Services
  If you use Oracle Rest Data Services (ORDS), then you must authorize proxy users.
- Oracle APEX and Application Objects Protected by Oracle Database Vault
  Objects that are protected by Oracle Database Vault realms and command rules are still protected after you have integrated Oracle APEX.
- Troubleshooting the Oracle APEX and Database Vault Integration
  If you have problems with the integration of Oracle APEX and Database Vault, then you can diagnose these problems using tracing and Oracle Database Vault simulation mode.

11.7.1 About Integrating Oracle Database Vault with Oracle APEX

Oracle APEX is Oracle's primary tool for developing Web applications with SQL and PL/SQL.

You can enable Oracle Database Vault to protect the applications that are developed on Oracle APEX. To use Oracle APEX in an Oracle Database Vault-enabled database, you must install Oracle APEX on the Oracle Database Vault-enabled server, and then perform the necessary authorizations in the appropriate pluggable databases (PDBs).

11.7.2 Installing or Upgrading Oracle APEX with Oracle Database Vault Enabled

When Oracle Database Vault is enabled, additional privileges are required to install or upgrade Oracle APEX.

1. Log in to the root of the Oracle Database Vault-enabled database as a user with the `DV_OWNER` role.
2. Grant the `DV_PATCH_ADMIN` role to `SYS` in all pluggable databases (PDBs).

```
GRANT DV_PATCH_ADMIN TO SYS CONTAINER=ALL;
```

3. Complete the installation of Oracle APEX as directed by *Oracle APEX Installation Guide*.

   If this installation includes Oracle Rest Data Services (ORDS), then see Authorizing Oracle APEX to Proxy Users for Oracle Rest Data Services.

4. As a user with the `DV_OWNER` role, in the container database revoke the `DV_PATCH_ADMIN` role from `SYS` from all PDBs.

```
REVOKE DV_PATCH_ADMIN FROM SYS CONTAINER=ALL;
```

11.7.3 Authorizing the Oracle APEX Schema for Oracle Database Vault Activities

You must add the Oracle APEX schema (for example, `APEX_SCHEMA`) to Oracle Database Vault realms and authorizations that are required by Oracle APEX.

1. Connect to the pluggable database (PDB) of the Oracle Database Vault-enabled database as a user with the `DV_OWNER` role.
2. Authorize the Oracle APEX schema for the following realms, policies, and rules.

   For example, for an Oracle APEX schema called `APEX_SCHEMA`:

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
 REALM_NAME      => 'Oracle Default Schema Protection Realm',
 GRANTEE         => 'APEX_SCHEMA',
 RULE_SET_NAME   => NULL,
 AUTH_OPTIONS    => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/

BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
 REALM_NAME      => 'Oracle System Privilege and Role Management Realm',
 GRANTEE         => 'APEX_SCHEMA',
 RULE_SET_NAME   => NULL,
 AUTH_OPTIONS    => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/

BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
 REALM_NAME     => 'Oracle Default Component Protection Realm',
 GRANTEE        => 'APEX_SCHEMA',
 RULE_SET_NAME  => NULL,
 AUTH_OPTIONS   => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/

BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
 REALM_NAME     => 'Database Vault Account Management',
 GRANTEE        => 'APEX_SCHEMA',
 RULE_SET_NAME  => NULL,
```

```
 AUTH_OPTIONS    => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/

EXEC DBMS_MACADM.UPDATE_POLICY_STATE('Oracle Account Management
Controls',DBMS_MACADM.G_PARTIAL);
BEGIN
 DBMS_MACADM.UPDATE_RULE('Is User Manager'
,'DVSYS.DBMS_MACUTL.USER_HAS_ROLE_VARCHAR(''DV_ACCTMGR'',SYS_CONTEX
T(''userenv'',''current_user'')) = ''Y''');
END;
/

BEGIN
 DBMS_MACADM.UPDATE_RULE('Is Alter DVSYS Allowed'
,'DVSYS.DBMS_MACADM.IS_ALTER_USER_ALLOW_VARCHAR(SYS_CONTEXT(''useren
v'',''current_user'')) = ''Y''');
END;
/
```

11.7.4 Authorizing Oracle APEX to Use Oracle Scheduler

Oracle APEX uses Oracle Scheduler and must be authorized to continue to do so.

1. Connect the pluggable database (PDB) of the Oracle Database Vault-enabled database as a user with the `DV_OWNER` role.
2. Authorize the `SYS` user to use its scheduler privileges on the Oracle APEX schema.

   For example, for an Oracle APEX schema named `APEX_SCHEMA`:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('SYS','APEX_SCHEMA');
   ```

3. Authorize the Oracle APEX user to use its job scheduler privileges.

   ```
   EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('APEX_SCHEMA','%');
   ```

Related Topics

- Using Oracle Scheduler with Oracle Database Vault
  Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific authorization.

11.7.5 Authorizing Oracle APEX to Perform DDL Tasks

You must authorize the Oracle APEX schema to use its DDL privileges on objects that it has access to but may be subject to additional Oracle Database Vault controls

1. Connect the pluggable database (PDB) of the Oracle Database Vault-enabled database as a user with the `DV_OWNER` role.
2. Authorize the Oracle APEX schema to use its DDL privileges

For example, for an Oracle APEX schema named `APEX_SCHEMA`:

```
EXEC DBMS_MACADM.AUTHORIZE_DDL('APEX_SCHEMA','%');
```

Related Topics

- Performing DDL Operations in Oracle Database Vault
  Data Definition Language (DDL) operations in Oracle Database Vault can be affected by situations such as schema ownership and patch upgrades.

11.7.6 Authorizing Oracle APEX to Perform Information Lifecycle Maintenance Tasks

You must authorize the Oracle APEX schema to perform maintenance tasks.

1. Connect the pluggable database (PDB) of the Oracle Database Vault-enabled database as a user with the `DV_OWNER` role.
2. Authorize the Oracle APEX schema to use its DDL privileges.

   For example, for an Oracle APEX schema named `APEX_SCHEMA`:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER('APEX_SCHEMA','%');
   ```

Related Topics

- Using Information Lifecycle Management with Oracle Database Vault
  Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.

11.7.7 Authorizing Oracle APEX to Proxy Users for Oracle Rest Data Services

If you use Oracle Rest Data Services (ORDS), then you must authorize proxy users.

1. Connect the pluggable database (PDB) of the Oracle Database Vault-enabled database as a user with the `DV_OWNER` role.
2. Find the existing proxy users and database users that they can proxy as.

   ```
   SELECT PROXY, CLIENT FROM PROXY_USERS;
   ```

3. Authorize the proxying of users.

   ```
   EXEC DBMS_MACADM.AUTHORIZE_PROXY_USER('proxy','client');
   ```

4. Repeat this step for each of the combinations listed in the `PROXY_USERS` query that you performed.

Related Topics

- Using Information Lifecycle Management with Oracle Database Vault
  Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.

11.7.8 Oracle APEX and Application Objects Protected by Oracle Database Vault

Objects that are protected by Oracle Database Vault realms and command rules are still protected after you have integrated Oracle APEX.

The same privileges and authorizations must be met before Oracle Database Vault will grant access to these objects. For example, if you create an Oracle APEX workspace that requires access to the `HR` schema objects, and there is an Oracle Database Vault realm protected the `HR` schema objects, then the workspace will be required to have authorization to access the realm.

Related Topics

- Configuring Realms
  You can create a realm around database objects to protect them, and then set authorizations to control user access to this data.
- Configuring Command Rules
  You can create command rules or use the default command rules to protect DDL and DML statements.

11.7.9 Troubleshooting the Oracle APEX and Database Vault Integration

If you have problems with the integration of Oracle APEX and Database Vault, then you can diagnose these problems using tracing and Oracle Database Vault simulation mode.

- **Tracing:** Trace files enable you to track the Oracle Database Vault database instance for server and background process events. Use trace file to find out if the Oracle Database Vault policy authorization succeeded or failed. They are also useful for resolving issues such as bugs and other unexpected behavior.
- **Simulation mode:** You can use simulation mode to capture violations in a simulation log instead of blocking SQL execution by Oracle Database Vault realms and command rules. Oracle Database Vault stores these errors in a central location so that you can easily analyze them.

Related Topics

- Using Trace Files to Diagnose Oracle Database Vault Events
  Trace files, which the database generates, capture important information to help you debug errors.
- Using Simulation Mode for Logging Realm and Command Rule Activities
  Simulation mode writes violations to the simulation log instead of preventing SQL execution to quickly test new and modified Oracle Database Vault controls.

12

DBA Operations in an Oracle Database Vault Environment

Database administrators can perform operations in an Oracle Database Vault environment, such as using Database Vault with products such as Oracle Data Pump.

- Performing DDL Operations in Oracle Database Vault
  Data Definition Language (DDL) operations in Oracle Database Vault can be affected by situations such as schema ownership and patch upgrades.
- Using SQL Firewall with Oracle Database Vault
  You can authorize SQL Firewall users to work in a Database Vault environment.
- Using Oracle Database Vault with Oracle Enterprise Manager
  Oracle Database Vault administrators can perform tasks in Oracle Enterprise Manager Cloud Control such as propagating polices to other databases.
- Using Oracle Data Pump with Oracle Database Vault
  Database administrators can authorize Oracle Data Pump users to work in a Database Vault environment.
- Using Oracle Scheduler with Oracle Database Vault
  Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific authorization.
- Using Information Lifecycle Management with Oracle Database Vault
  Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.
- Using Oracle Database Replay with Oracle Database Vault
  Database administrators can authorize Oracle Database Replay users to work in a Database Vault environment.
- Running Preprocessor Programs with Oracle Database Vault
  Users who run preprocessor programs through external tables must have Oracle Database Vault-specific authorization.
- Using Database Vault Operations Control to Restrict Multitenant Common User Access to Local PDB Data
  You can control PDB access by CDB root common users, such as infrastructure database administrators.
- Preventing Multitenant Local Users from Blocking Common Operations
  You can prevent multitenant local users from blocking common operations when they attempt to create Oracle Database Vault protections on common user objects.
- Oracle Recovery Manager and Oracle Database Vault
  You can use Recovery Manager (RMAN) in an Oracle Database Vault environment.
- Privileges for Using XStream with Oracle Database Vault
  If you want to use XStream in an Oracle Database Vault environment, then you must have the appropriate privileges.

- Privileges for Using Oracle GoldenGate with Oracle Database Vault
  If you want to use Oracle GoldenGate in an Oracle Database Vault environment, then you must have the appropriate privileges.
- Using Data Masking in an Oracle Database Vault Environment
  You must have the correct authorization to perform data masking in an Oracle Database Vault environment.
- Using Oracle Database Auditing with Oracle Database Vault
  You grant and revoke authorizations to manage and view audit records in an Oracle Database Vault environment.
- Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB
  You can convert a standalone Oracle Database database from release 12c through 19c to a PDB, and then plug this PDB into a CDB.
- Using the ORADEBUG Utility with Oracle Database Vault
  The `ORADEBUG` utility is used primarily by Oracle Support to diagnose problems that may arise with an Oracle database.
- Performing Patch Operations in an Oracle Database Vault Environment
  User `SYS` must have the `DV_PATCH_ADMIN` role to perform a patch operations on an Oracle Database Vault-enabled database.

12.1 Performing DDL Operations in Oracle Database Vault

Data Definition Language (DDL) operations in Oracle Database Vault can be affected by situations such as schema ownership and patch upgrades.

- Restrictions on Performing DDL Operations in Oracle Database Vault
  Depending on the Oracle Database Vault configuration, DDL operations may be restricted and require DDL authorizations in an Oracle Database Vault environment.
- Impact of the DV_PATCH_ADMIN Role on DDL Operations
  Object owners and users who have been granted the `DV_PATCH_ADMIN` role are exempt from the DDL authorization requirement.
- Impact of Upgrades from Releases 21c and Earlier on DDL Operations
  If you upgrade Oracle Database Vault from release 21c or earlier, you may need to change the DDL authorizations.
- Impact of the Removal of the DDL Default Authorization of ('%', '%')
  The DDL default authorization of (`'%', '%'`) enables a user to perform DDL operations on any schema without explicit DDL authorizations.

12.1.1 Restrictions on Performing DDL Operations in Oracle Database Vault

Depending on the Oracle Database Vault configuration, DDL operations may be restricted and require DDL authorizations in an Oracle Database Vault environment.

Specifically, a user is required to have DDL authorization to perform DDL operations on a schema that has any of the following characteristics:

- The schema is an owner of objects that are protected by enabled realms.
- The schema is authorized to any enabled realm directly or through roles.

- The schema is granted object privileges directly or through roles on objects that are protected by enabled realms.
- The schema is granted any Oracle Database Vault roles directly or through roles.

Object owners and users who have granted the `DV_PATCH_ADMIN` role are exempt from the DDL authorization requirement. You can authorize a user to perform DDL operations on a specific schema by using the `DBMS_MACADM.AUTHORIZE_DDL` procedure. Note, however, that DDL authorization does not enable the grantee to perform DDL operations on a realm protected object or schema. To enable such operations, you must authorize the user for the realm. To find information about users who have been granted this authorization, query the `DBA_DV_DDL_AUTH` data dictionary view.

If Oracle Database Vault is upgraded from a previous release older than Oracle Database 21c, then the default DDL authorization of (`%`, `%`) may exist, and it would enable users to perform DDL operations on any schema without explicit DDL authorizations. For better security, Oracle recommends that you remove the default DDL authorization by running `DBMS_MACADM.UNAUTHORIZE_DDL('%', '%')` and grant required DDL authorizations only to users who need to perform DDL operations.

Related Topics

- AUTHORIZE_DDL Procedure
  The `AUTHORIZE_DDL` procedure grants a user authorization to run Data Definition Language (DDL) statements on the specified schema.

12.1.2 Impact of the DV_PATCH_ADMIN Role on DDL Operations

Object owners and users who have been granted the `DV_PATCH_ADMIN` role are exempt from the DDL authorization requirement.

You can authorize a user to perform DDL operations on a specific schema by using the `DBMS_MACADM.AUTHORIZE_DDL` procedure. Note, however, that DDL authorization does not allow the grantee to perform DDL operations on a realm-protected object or schema. To allow such operations, you must authorize the user for the realm. To find information about users who have been granted this authorization, query the `DBA_DV_DDL_AUTH` data dictionary view.

Related Topics

- AUTHORIZE_DDL Procedure
  The `AUTHORIZE_DDL` procedure grants a user authorization to run Data Definition Language (DDL) statements on the specified schema.

12.1.3 Impact of Upgrades from Releases 21c and Earlier on DDL Operations

If you upgrade Oracle Database Vault from release 21c or earlier, you may need to change the DDL authorizations.

The default DDL authorization of (`'%', '%'`) may exist, which enables users to perform DDL operations on any schema without explicit DDL authorizations. For better security, Oracle recommends that you remove the default DDL authorization by running `DBMS_MACADM.UNAUTHORIZE_DDL('%', '%')` and then grant required DDL authorizations only to users who need to perform DDL operations.

Related Topics

- UNAUTHORIZE_DDL Procedure
  The `UNAUTHORIZE_DDL` procedure revokes authorization from a user who was granted authorization to run DDL statements through the `DBMS_MACDM.AUTHORIZE_DDL` procedure.
- AUTHORIZE_DDL Procedure
  The `AUTHORIZE_DDL` procedure grants a user authorization to run Data Definition Language (DDL) statements on the specified schema.

12.1.4 Impact of the Removal of the DDL Default Authorization of ('%', '%')

The DDL default authorization of (`'%', '%'`) enables a user to perform DDL operations on any schema without explicit DDL authorizations.

This default DDL authorization, which has been in place since DDL authorization was introduced in Oracle Database release 12.1, was to prevent any undesirable disruption due to unexpected DDL failures in the Oracle Database Vault environment. From Database Vault release 21c, however, there is no default DDL authorization, and the existing default DDL authorization of (`'%', '%'`) is removed when Database Vault is upgraded to 21c or later. To prevent any problems, you need to identify and authorize trusted database users for DDL operations or optionally re-authorize (`'%', '%'`) so that every user is allowed to perform DDL operations without explicit authorization. For better security, Oracle recommends that only trusted users are authorized for DDL operations.

Related Topics

- UNAUTHORIZE_DDL Procedure
  The `UNAUTHORIZE_DDL` procedure revokes authorization from a user who was granted authorization to run DDL statements through the `DBMS_MACDM.AUTHORIZE_DDL` procedure.
- AUTHORIZE_DDL Procedure
  The `AUTHORIZE_DDL` procedure grants a user authorization to run Data Definition Language (DDL) statements on the specified schema.

12.2 Using SQL Firewall with Oracle Database Vault

You can authorize SQL Firewall users to work in a Database Vault environment.

- About Using SQL Firewall with Oracle Database Vault
  Database administrators who use SQL Firewall in an Oracle Database Vault environment must have Database Vault-specific authorization.
- Authorizing Users for SQL Firewall in Database Vault
  You can authorize a database administrator to use SQL Firewall in an Oracle Database Vault environment.
- Revoking SQL Firewall Authorization from Users
  You can revoke SQL Firewall authorization from a database administrator in an Oracle Database Vault environment.

12.2.1 About Using SQL Firewall with Oracle Database Vault

Database administrators who use SQL Firewall in an Oracle Database Vault environment must have Database Vault-specific authorization.

In addition to the Database Vault-specific authorization, these users must have the standard SQL Firewall privileges (the `ADMINISTER SQL FIREWALL` system privilege or the `SQL_FIREWALL_ADMIN` role). You can configure the authorization to prevent SQL Firewall from blocking users with `DV_OWNER` and `DV_ACCTMGR`. These users must be database users only. When Database Vault operations control is enabled, common users will be blocked from using SQL Firewall (that is, the `DBMS_SQL_FIREWALL` procedures for managing captures and allow-lists) on local users unless the common users are included in the exception list.

You can check a user's authorizations for using SQL Firewall in an Oracle Database Vault environment by querying the `DBA_DV_SQL_FIREWALL_AUTH` data dictionary view.

Related Topics

- Using Database Vault Operations Control to Restrict Multitenant Common User Access to Local PDB Data
  You can control PDB access by CDB root common users, such as infrastructure database administrators.

12.2.2 Authorizing Users for SQL Firewall in Database Vault

You can authorize a database administrator to use SQL Firewall in an Oracle Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Ensure that the database user to whom you want to grant authorization has been granted the `ADMINISTER SQL FIREWALL` system privilege or the `SQL_FIREWALL_ADMIN` role.

   You can query the `DBA_SYS_PRIVS` data dictionary view to find the privileges that the user has been granted, and the `DBA_ROLES` view for the user's roles.
3. Grant this user Oracle Database Vault authorization to use SQL Firewall.

   For example, to authorize the user `PFITCH`:

   ```
   BEGIN
     DBMS_MACADM.AUTHORIZE_SQL_FIREWALL (
       uname             => 'PFITCH',
       manage_dv_admins  => 'Y');
   END;
   /
   ```

   In this specification, `manage_dv_admins` controls whether the authorized user is allowed to use SQL Firewall on user accounts that have been granted the `DV_OWNER` or `DV_ACCTMGR` role. A setting of `Y` enables the user to use SQL Firewall for users who have been granted either of these roles. A setting of `N` (the default) prevents the user from using SQL Firewall for these users. The SQL Firewall procedures (that is, the `DBMS_SQL_FIREWALL` package procedures for managing captures and allow-lists) will raise an error if the target user is granted the `DV_OWNER` or `DV_ACCTMGR` role and the current user does not have SQL Firewall authorization with `manage_dv_admins` set to `Y`.

4. To check the authorization of the user, query the `DBA_DV_SQL_FIREWALL_AUTH` data dictionary view.

   For example:

   ```
   SELECT * FROM DBA_DV_SQL_FIREWALL_AUTH WHERE UNAME = 'PFITCH';
   ```

Related Topics

- AUTHORIZE_SQL_FIREWALL Procedure
  The `AUTHORIZE_SQL_FIREWALL` procedure grants a user authorization to use SQL Firewall.

12.2.3 Revoking SQL Firewall Authorization from Users

You can revoke SQL Firewall authorization from a database administrator in an Oracle Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Check the current SQL Firewall authorization for the user from whom you want to revoke authorization.

   For example:

   ```
   SELECT * FROM DBA_DV_SQL_FIREWALL_AUTH
   WHERE UNAME = 'PFITCH';
   ```

   Make a note of the output. For example:

   ```
   GRANTEE  CAN_MANAGE_DV_ADMINS
   -------  --------------------
   PFITCH   Y
   ```

3. Revoke the Database Vault authorization to use SQL Firewall from the user. Ensure that the settings that you use match the settings that were previously used to authorize the user. Otherwise, the revocation will fail with an `ORA-47896: SQL Firewall authorization for Oracle Database Vault to PFITCH is not found` error.

   For example, if the user's `manage_dv_admins` setting was `Y`, then you must use that setting when you revoke the SQL Firewall authorization from the user.

   ```
   BEGIN
     DBMS_MACADM.UNAUTHORIZE_SQL_FIREWALL (
       uname             => 'PFITCH',
       manage_dv_admins  => 'Y');
   END;
   /
   ```

Related Topics

- UNAUTHORIZE_SQL_FIREWALL Procedure
  The `UNAUTHORIZE_SQL_FIREWALL` procedure revokes a user's authorization to use SQL Firewall.

12.3 Using Oracle Database Vault with Oracle Enterprise Manager

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

Using Oracle Database Vault with Oracle Enterprise Manager Express is not supported. Oracle Enterprise Manager Cloud Control provides the pages for Oracle Database Vault.

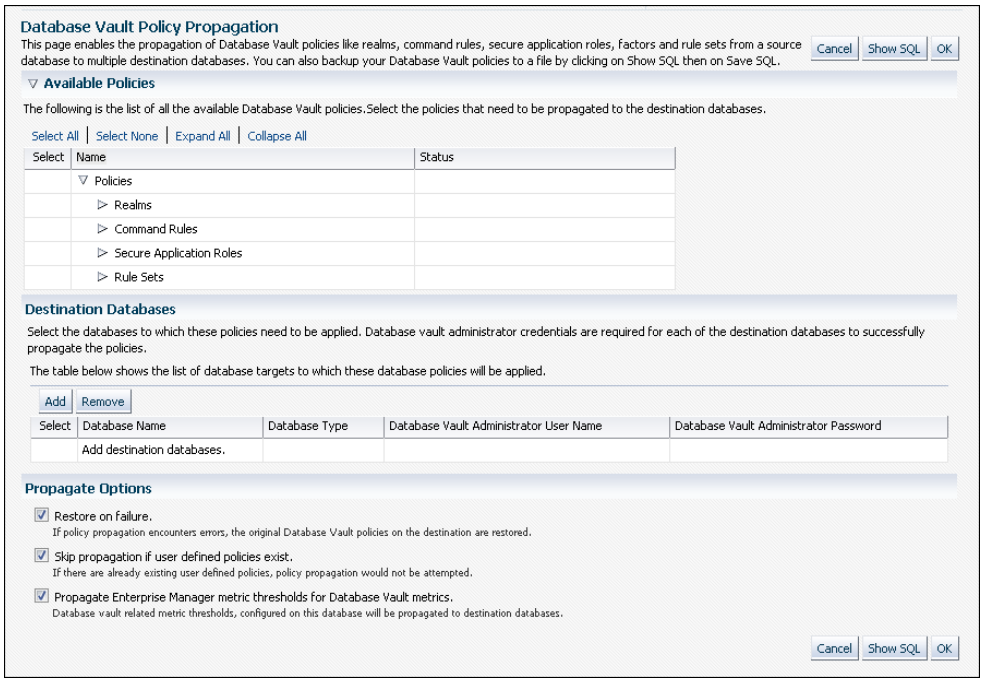
- Propagating Oracle Database Vault Configurations to Other Databases
  You can propagate Database Vault configurations (such as a realm configuration) to other Database Vault-protected databases.
- Enterprise Manager Cloud Control Alerts for Oracle Database Vault Policies
  To view Oracle Database Vault alerts, you must be granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.
- Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control
  From the Database Vault home page, you can find information about violations.

12.3.1 Propagating Oracle Database Vault Configurations to Other Databases

You can propagate Database Vault configurations (such as a realm configuration) to other Database Vault-protected databases.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Database Vault home page, under Database Vault Policy Propagation, select **Database Vault Policy Propagation**.

   The Available Policies area in the Policy Propagation subpage lists a summary of the Oracle Database Vault configurations that were created for the current database: that is, configurations that were created for realms, command rules, rule sets, and secure application roles. It does not list the Oracle Database Vault policies that were introduced in Oracle Database release 12*c* (12.2). From here, you can propagate these configurations to another database.
3. Under Available Policies, select each configuration that you want to propagate to another database.



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

      This feature applies the Database Vault administrator's user name and password to all of the selected destination databases.
   b. Select each database to which you want to propagate the configurations.
   c. Enter the Database Vault administrator user name and password for each database.
   d. Click the **Apply** button.
7. In the Propagate Options page, select from the following options.

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

   - **Restore on failure:** If the propagation operations encounters errors, then the propagation is rolled back. That is, the original policies on the destination database are restored. If you do not select this option, then the policy propagation on the destination database continues and ignores any errors.
   - **Skip propagation if user defined policies exist:** If the destination databases already have the user-defined configurations, then the propagation operation is not attempted. If you do not select this option, then regardless of whether user-defined policies exist on the destination database, all the existing

configurations are cleared, and the configurations from the source database are applied to the destination database.

- **Propagate Enterprise Manager metric thresholds for database vault metrics:** If the source database has Oracle Database Vault metric thresholds set, then these thresholds are also propagated to the destination databases. If you do not select this option, then only configurations are propagated and not the Oracle Database Vault thresholds.

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

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

12.3.2 Enterprise Manager Cloud Control Alerts for Oracle Database Vault Policies

To view Oracle Database Vault alerts, you must be granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role.

The alerts are as follows:

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

12.3.3 Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control

From the Database Vault home page, you can find information about violations.

These violations are as follows:

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

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

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.

12.4 Using Oracle Data Pump with Oracle Database Vault

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

- About Using Oracle Data Pump with Oracle Database Vault
  Database administrators who use Oracle Data Pump in an Database Vault environment must have Database Vault-specific authorization to export and import data.
- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.
- Authorizing Users or Roles for Data Pump Transportable Export and Import Operations
  You can grant authorization levels for users who must perform Oracle Data Pump transportable operations, either directly or through a role.
- Guidelines for Exporting or Importing Data in a Database Vault Environment
  After you grant the Oracle Data Pump database administrator the proper authorization, this user can perform any export or import operations that are necessary.

12.4.1 About Using Oracle Data Pump with Oracle Database Vault

Database administrators who use Oracle Data Pump in an Database Vault environment must have Database Vault-specific authorization to export and import data.

This type of user must have Database Vault privileges in addition to the standard Oracle Data Pump privileges. If these users want to perform Oracle Data Pump transportable tablespace operations, then they must have special authorization. You can check a user's authorizations for using Data Pump in an Oracle Database Vault environment by querying the `DBA_DV_DATAPUMP_AUTH` data dictionary view. You can grant this authorization to either individual users or to database roles.

12.4.2 Authorizing Users or Roles for Data Pump Regular Export and Import Operations

You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

- About Authorizing Users or Roles for Oracle Data Pump Regular Operations
  Users who have Oracle Data Pump authorization can perform regular Oracle Data Pump operations in a Database Vault environment.

- Levels of Database Vault Authorization for Oracle Data Pump Regular Operations
  Oracle Database Vault provides several levels of authorization required for Oracle Data Pump regular operations in a Database Vault environment.
- Authorizing Users or Roles for Oracle Data Pump Regular Operations in Database Vault
  You can authorize a database administrator or a role to use Data Pump for regular operations in an Oracle Database Vault environment.
- Revoking Oracle Data Pump Authorization from Users or Roles
  You can revoke authorization from the database administrator or role who is using Oracle Data Pump for regular operations.

12.4.2.1 About Authorizing Users or Roles for Oracle Data Pump Regular Operations

Users who have Oracle Data Pump authorization can perform regular Oracle Data Pump operations in a Database Vault environment.

You can perform the following types of Oracle Data Pump authorizations:

- Authorizing the user or role to be able to import protected schemas and objects
- Authorizing the user or role to be able to perform following activities that can take place during the import operation: the creation of users, the grant of Oracle Database Vault-protected roles and system privileges, the grant of specific Oracle Database roles, and the grant of Oracle Database system privileges

> **Note:**
>
> Full level Data Pump authorization enables these users to perform transportable export and import operations as well.

Related Topics

- Authorizing Users or Roles for Data Pump Transportable Export and Import Operations
  You can grant authorization levels for users who must perform Oracle Data Pump transportable operations, either directly or through a role.

12.4.2.2 Levels of Database Vault Authorization for Oracle Data Pump Regular Operations

Oracle Database Vault provides several levels of authorization required for Oracle Data Pump regular operations in a Database Vault environment.

Table 12-1 describes these levels.

Table 12-1 Levels of Authorization for Oracle Data Pump Regular Operations

Scenario	Authorization Required
A database administrator wants to import data into another schema.	You must grant this user (or a role) the `BECOME USER` system privilege and the `IMP_FULL_DATABASE` role.[1] To find the privileges a user has been granted, query the `USER_SYS_PRIVS` data dictionary view.

Table 12-1 (Cont.) Levels of Authorization for Oracle Data Pump Regular Operations

Scenario	Authorization Required
A database administrator wants to export or import data in a schema that has no Database Vault protection.	You only need to grant this user (or a role) the standard Oracle Data Pump privileges, which are the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles. If the user wants to import data, grant this user the `BECOME USER` system privilege.
A database administrator wants to export or import data in a protected schema.	In addition to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles, you must grant this user (or a role) Database Vault-specific authorization by using the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure. This authorization applies to both the `expdp` and `impdp` utilities. Later on, you can revoke this authorization by using the `DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER` procedure. If the user wants to import data, also grant this user the `BECOME USER` system privilege.
A database administrator wants to export or import the contents of an entire database.	In addition to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles and the authorization granted by the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure, you must grant this user (or a role) the `DV_OWNER` role. If the user wants to import data, grant this user the `BECOME USER` system privilege.

[1] The `BECOME USER` privilege is part of the `IMP_FULL_DATABASE` role by default, but in an Oracle Database Vault environment, this privilege is revoked.

12.4.2.3 Authorizing Users or Roles for Oracle Data Pump Regular Operations in Database Vault

You can authorize a database administrator or a role to use Data Pump for regular operations in an Oracle Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Ensure that the user or role to whom you want to grant authorization has been granted the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles, which are required for using Oracle Data Pump.

   ```
   SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE LIKE
   '%FULL%';
   ```

3. Grant this user or role Oracle Database Vault authorization to import protected schemas and objects.

   For example, to authorize the Data Pump user `DP_MGR` to export and import objects for the database table `EMPLOYEES`:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR', 'EMPLOYEES');
   ```

   To restrict `DP_MGR`'s activities to a specific schema, you would enter the following procedure:

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

   To authorize users who have been granted the `DP_MGR_ROLE` role to export and import objects for the entire database, enter the following:

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

After you run the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure, you can check the authorization of the user or role by querying the `DBA_DV_DATAPUMP_AUTH` data dictionary view.

If you granted the user or role full authorization (using `%` for the `schema`, `object`, `type`, and `action` parameters), then you can bypass the next step. However, if the authorization is only for a specific schema (for example, `schema` is set to `HR` and the remaining parameters are still set to `%`), then you must perform the next step.

4. If necessary, grant the user or role authorization to perform the following activities during the import operation:
   a. Creating users during the import. For example:

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

   b. Granting Oracle Database Vault-protected roles and system privileges during the import. For example:

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

   c. Granting a specific role during the import. For example:

      ```
      EXEC DBMS_MACADM.AUTH_DATAPUMP_GRANT_ROLE('DP_MGR', 'DBA');
      ```

   d. Granting system privileges during the import. For example:

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

5. If the user or role must export the entire database, then grant them the `DV_OWNER` role.

   For example, for a role:

   ```
   GRANT DV_OWNER TO DP_MGR_ROLE;
   ```

Related Topics

- About Authorizing Users or Roles for Oracle Data Pump Regular Operations
  Users who have Oracle Data Pump authorization can perform regular Oracle Data Pump operations in a Database Vault environment.
- AUTHORIZE_DATAPUMP_USER Procedure
  The `AUTHORIZE_DATAPUMP_USER` procedure authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled.
- DBA_DV_DATAPUMP_AUTH View
  The `DBA_DV_DATAPUMP_AUTH` data dictionary view lists the authorizations for using Oracle Data Pump in an Oracle Database Vault environment.

12.4.2.4 Revoking Oracle Data Pump Authorization from Users or Roles

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

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

```
REVOKE DV_OWNER FROM DP_MGR_ROLE;
```

2. Query the `DBA_DV_DATAPUMP_AUTH` data dictionary view to find the users or roles that have been granted Oracle Data Pump authorizations.

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

3. Use the information you gathered from the preceding step to build the `DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER` command.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR', 'EMPLOYEES');
   ```

   Ensure that this unauthorization complements the original authorization action. In other words, if you originally gave `DP_MGR` authorization over the entire database, then the following commands will not work:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR');
   EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR', 'EMPLOYEES');
   ```

4. If you authorized the user or role to perform user creation or other activities during the import operation, then revoke these.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTH_DATAPUMP_CREATE_USER('DP_MGR');
   EXEC DBMS_MACADM.UNAUTH_DATAPUMP_GRANT('DP_MGR');
   EXEC DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_ROLE('DP_MGR', 'DBA');
   EXEC DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_SYSPRIV('DP_MGR');
   ```

   You can find the user's authorizations by querying the `DBA_DV_DATAPUMP_AUTH` data dictionary view.

Related Topics

- UNAUTHORIZE_DATAPUMP_USER Procedure
  The `UNAUTHORIZE_DATAPUMP_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_DATAPUMP_USER` procedure.
- DBA_DV_DATAPUMP_AUTH View
  The `DBA_DV_DATAPUMP_AUTH` data dictionary view lists the authorizations for using Oracle Data Pump in an Oracle Database Vault environment.

12.4.3 Authorizing Users or Roles for Data Pump Transportable Export and Import Operations

You can grant authorization levels for users who must perform Oracle Data Pump transportable operations, either directly or through a role.

- About Authorizing Users for Oracle Data Pump Transportable Operations
  You can grant users (either directly or through a role) different levels of transportable operation authorization.
- Levels of Database Vault Authorization for Data Pump Transportable Operations
  Oracle Database Vault provides levels of authorization required for users who must perform export and import transportable operations in a Database Vault environment.

- Authorizing Users or Roles for Data Pump Transportable Operations in Database Vault
  You can authorize users or roles to perform Oracle Data Pump transportable export or import operations in a Database Vault environment.
- Revoking Transportable Tablespace Authorization from Users or Roles
  You can revoke authorization from the database administrator who is using Data Pump.

12.4.3.1 About Authorizing Users for Oracle Data Pump Transportable Operations

You can grant users (either directly or through a role) different levels of transportable operation authorization.

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

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

12.4.3.2 Levels of Database Vault Authorization for Data Pump Transportable Operations

Oracle Database Vault provides levels of authorization required for users who must perform export and import transportable operations in a Database Vault environment.

Table 12-2 describes these levels.

Table 12-2 Levels of Authorization for Oracle Data Pump Transporatable Operations

Scenario	Authorization Required
A database administrator wants to transportable export a tablespace or table that has no Database Vault protection.	You only need to grant this user (or a role) the standard Oracle Data Pump privileges, which are the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles.
A database administrator wants to transportable export a tablespace where there is Database Vault protection (for example, realm or command rule for a table object residing on that tablespace).	In addition to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles, you must grant this user (or a role) Database Vault-specific transportable tablespace authorization by using the `DBMS_MACADM.AUTHORIZE_TTS_USER` procedure. Later on, you can revoke this authorization by using the `DBMS_MACADM.UNAUTHORIZE_TTS_USER` procedure. Remember that users who have been granted full database level Oracle Data Pump authorization (through the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure) can perform these operations as well.
A database administrator wants to transportable export a table within a tablespace where there is Database Vault protection (for example, a realm or command rule for a table object residing on the tablespace that contains the table to be exported).	In addition to the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles, you must grant this user (or a role) Database Vault-specific transportable tablespace authorization for the tablespace that contains the table to be exported by using the `DBMS_MACADM.AUTHORIZE_TTS_USER` procedure. Remember that users who have been granted full database level Oracle Data Pump authorization (from the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure) can perform these operations as well.

Table 12-2 (Cont.) Levels of Authorization for Oracle Data Pump Transporatable Operations

Scenario	Authorization Required
A database administrator wants to transportable export the contents of an entire database.	In addition to the DV_OWNER, EXP_FULL_DATABASE, and IMP_FULL_DATABASE roles, you must grant this user (or a role) Database Vault-specific full database level Oracle Data Pump authorization by using the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure. You do not need to run the DBMS_MACADM.AUTHORIZE_TTS_USER procedure for this user.
A database administrator wants to use a network link to transportable import a tablespace or a table that has no Database Vault protection.	In addition to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles for both the database administrator and the connecting user, you must grant the connecting user (or a role) specified in the network link the DV_DATAPUMP_NETWORK_LINK role.
A database administrator wants to use a network link to transportable import a tablespace where there is Database Vault protection (for example, realm or command rule for a table object residing on that tablespace)	In addition to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles, you must grant the connecting user (or a role) specified in the network link the Database Vault-specific transportable tablespace authorization for that tablespace by using the DBMS_MACADM.AUTHORIZE_TTS_USER procedure. You must also grant the connecting user the DV_DATAPUMP_NETWORK_LINK role. Remember that users that have been granted Database Vault-specific full database level Oracle Data Pump authorization (through the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure) can perform these operations.
A database administrator wants to use a network link to import a table within a transportable tablespace where there is Database Vault protection (for example, realm or command rule for a table object residing on the tablespace that contains the table to be exported)	In addition to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles, you must grant the connecting user (or a role) the Database Vault-specific transportable tablespace authorization for the tablespace that contains the table to be exported by using the DBMS_MACADM.AUTHORIZE_TTS_USER procedure. You also must grant the connecting user (or a role) specified in the network link the DV_DATAPUMP_NETWORK_LINK role. Remember that users who have been granted Database Vault-specific full database level Oracle Data Pump authorization (through the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure) can perform the operations.
A database administrator wants to use a network link to transportable import the contents of an entire database.	In addition to the DV_OWNER role, you must grant the connecting user (or a role) Database Vault-specific full database level Oracle Data Pump authorization by using the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure. You do not need to run the DBMS_MACADM.AUTHORIZE_TTS_USER procedure for this user. You must also grant the connecting user (or a role) who is specified in the network link the DV_DATAPUMP_NETWORK_LINK role.

12.4.3.3 Authorizing Users or Roles for Data Pump Transportable Operations in Database Vault

You can authorize users or roles to perform Oracle Data Pump transportable export or import operations in a Database Vault environment.

1. Log into the PDB as a user who has been granted the DV_OWNER or DV_ADMIN role.

2. Ensure that the user or role to whom you want to grant authorization has been granted the `EXP_FULL_DATABASE` and `IMP_FULL_DATABASE` roles, which are required for using Oracle Data Pump.

```
SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS
 WHERE GRANTED_ROLE LIKE '%FULL%';
```

3. If the user wants to transportable export or use a network link to transportable import the contents of an entire database, then grant the full database level Oracle Data Pump authorization to the user or role by using the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure. Otherwise, bypass this step.

   For example:

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

4. If the user must have Database Vault-specific transportable tablespace authorization only, then grant this user or role this authorization.

   For example:

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

5. If the user who wants to perform a transportable import operation wants to use a network link to perform the operation, then grant this user or role the `DV_DATAPUMP_NETWORK_LINK` role.

   For example:

```
GRANT DV_DATAPUMP_NETWORK_LINK TO DP_MGR;
```

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

```
GRANT DV_OWNER TO DP_MGR;
```

Related Topics

- AUTHORIZE_TTS_USER Procedure
  The `AUTHORIZE_TTS_USER` procedure authorizes a user to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled.
- AUTHORIZE_DATAPUMP_USER Procedure
  The `AUTHORIZE_DATAPUMP_USER` procedure authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled.
- DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role
  The `DV_DATAPUMP_NETWORK_LINK` role is used for Data Pump import operations.

12.4.3.4 Revoking Transportable Tablespace Authorization from Users or Roles

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

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

```
REVOKE DV_OWNER FROM DP_MGR;
```

2. Query the `DBA_DV_TTS_AUTH` data dictionary view to find the users and roles that have been granted Oracle Data Pump authorizations.

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

3. Use the information you gathered from the preceding step to build the `DBMS_MACADM.UNAUTHORIZE_TTS_USER` statement.

   For example:

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

4. If the user had transportable exported or used a network link to transportable import the contents of an entire database, then revoke the full database level Oracle Data Pump authorization from the user or role.

   For example:

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

5. If the user who had performed a transportable import operation used a network link to perform the operation, then revoke the `DV_DATAPUMP_NETWORK_LINK` role from the user or role.

   For example:

   ```
   REVOKE DV_DATAPUMP_NETWORK_LINK FROM DP_MGR;
   ```

Related Topics

- UNAUTHORIZE_TTS_USER Procedure
  The `UNAUTHORIZE_TTS_USER` procedure removes from authorization users who had previously been granted the authorization to perform Oracle Data Pump transportable tablespace operations.
- UNAUTHORIZE_DATAPUMP_USER Procedure
  The `UNAUTHORIZE_DATAPUMP_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_DATAPUMP_USER` procedure.
- DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role
  The `DV_DATAPUMP_NETWORK_LINK` role is used for Data Pump import operations.

12.4.4 Guidelines for Exporting or Importing Data in a Database Vault Environment

After you grant the Oracle Data Pump database administrator the proper authorization, this user can perform any export or import operations that are necessary.

Before this user begins work, they should follow these guidelines:

- **Create a full backup of the database datafiles.** This way, if you or other users do not like the newly-imported data, then you easily can revert the database to its previous state. This guideline is especially useful if an intruder had managed to modify Oracle Data Pump exported data to use their own policies.
- **Decide how to handle exporting and importing multiple schemas or tables.** You cannot specify multiple schemas or tables in the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure, but you can use either of the following methods to accomplish this task:
  - Run the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure for each schema or table, and then specify the list of these objects in the `SCHEMAS` or `TABLES` parameter of the `EXPDP` and `IMPDP` utilities.
  - Perform a full database export or import operation. If so, see the next guideline.

- **When performing an export or import operation for an entire database, set the EXPDP or IMPDP FULL option to Y.** Remember that this setting will capture the `DVSYS` schema, so ensure that the user or role has that you have authorized been granted the `DV_OWNER` role.

Note the following:

- You cannot use the legacy `EXP` and `IMP` utilities with the direct path option (`direct=y`) if Oracle Database Vault is enabled.
- Users, either through a direct grant or a role grant, that have been granted Database Vault-specific Oracle Data Pump authorization through the `DBMS_MACADM.AUTHORIZE_DATAPUMP_USER` procedure or transportable tablespace authorization through the `DBMS_MACADM.AUTHORIZE_TTS_USER` procedure can export and import database objects, but they cannot perform other activities, such as `SELECT` queries on schema tables to which they normally do not have access. Similarly, users are not permitted to perform Data Pump operations on objects outside the designated data objects.
- You must grant the `DV_OWNER` role to users who must export or import an entire database, because a full database export requires access to the `DVSYS` schema, which stores the Oracle Database Vault policies. However, you cannot export the `DVSYS` schema itself. Data Pump only exports the protection definitions. The target database must have the `DVSYS` schema in it and Database Vault enabled before you can begin the import process.) Conversely, for a Data Pump import operation to apply the imported policies to the target database, it internally uses the `DBMS_MACADM` PL/SQL package, which in turn requires the Data Pump user to have the `DV_OWNER` role.

12.5 Using Oracle Scheduler with Oracle Database Vault

Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific authorization.

- About Using Oracle Scheduler with Oracle Database Vault
  The level of authorization that you must grant depends on the schema in which the administrator wants to perform a task.
- Granting a Job Scheduling Administrator Authorization for Database Vault
  You can authorize a user to schedule database jobs in a Database Vault environment.
- Revoking Authorization from Job Scheduling Administrators
  You can revoke authorization from a user for scheduling database jobs.

12.5.1 About Using Oracle Scheduler with Oracle Database Vault

The level of authorization that you must grant depends on the schema in which the administrator wants to perform a task.

Possible scenarios are as follows:

- **An administrator wants to schedule a job in their own schema.** An administrator who has been granted privileges to schedule database jobs can continue to do so without any Oracle Database Vault-specific authorizations, unless this schema is protected by a realm. In that case, ensure that this user is authorized to access the realm.
- **An administrator wants to run a job in another schema, but this job does not access any Oracle Database Vault realm or command rule protected object.** In this

case, this user only needs job related system privileges, not the Oracle Database Vault privileges.

- **An administrator wants to run a job under the schema of another user, including any schema in the database or a remote database.** If this job accesses an Oracle Database Vault realm or command rule protected object, then you must grant this user Database Vault-specific authorization by using the `DBMS_MACADM.AUTHORIZE_SCHEDULER_USER` procedure. This authorization applies to both background and foreground jobs. For background jobs, the authorization applies to the last user who created or modified the job. In addition, ensure that the schema owner (the protected schema in which the job is created) authorized to the realm.

  Later on, you can revoke this authorization by using the `DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER` procedure. If the schema is not protected by a realm, then you do not need to run the `DBMS_MACADM.AUTHORIZE_SCHEDULER_USER` procedure for the user.

Before you can enable or disable an Oracle Scheduler job that is protected by a realm, you must be authorized for that realm (using `DBMS_MACADM.ADD_AUTH_TO_REALM`), or you should have Oracle Scheduler authorization for the job owner schema (using `DBMS_MACADM.AUTHORIZE_SCHEDULER_USER`).

Related Topics

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

12.5.2 Granting a Job Scheduling Administrator Authorization for Database Vault

You can authorize a user to schedule database jobs in a Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant the necessary authorization.

2. Ensure that the user to whom you want to grant authorization has been granted system privileges to schedule database jobs.

   These privileges include any of the following: `CREATE JOB`, `CREATE ANY JOB`, `CREATE EXTERNAL JOB`, `EXECUTE ANY PROGRAM`, `EXECUTE ANY CLASS`, `MANAGE SCHEDULER`. The `DBA` and `SCHEDULER_ADMIN` roles provide these privileges; however, when Oracle Database Vault is enabled, the privileges are revoked from these roles.

   For example:

   ```
   SELECT GRANTEE, PRIVILEGE FROM DBA_SYS_PRIVS
    WHERE PRIVILEGE IN ('CREATE JOB', 'CREATE ANY JOB');
   ```

3. Grant this user Oracle Database Vault authorization.

   For example, to authorize the user `job_mgr` to schedule jobs for any schema in the database:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR');
   ```

Optionally, you can restrict `job_mgr`'s activities to a specific schema, as follows:

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

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

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

Related Topics

- AUTHORIZE_SCHEDULER_USER Procedure
  The `AUTHORIZE_SCHEDULER_USER` procedure grants a user authorization to schedule database jobs when Oracle Database Vault is enabled.
- DBA_DV_JOB_AUTH View
  The `DBA_DV_JOB_AUTH` data dictionary view lists the authorizations for using Oracle Scheduler in an Oracle Database Vault environment.

12.5.3 Revoking Authorization from Job Scheduling Administrators

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

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

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

2. Use the information you gathered from the preceding step to build the `DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER` command.

   For example:

```
EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR');
```

   Ensure that this unauthorization complements the original authorization action. In other words, if you originally gave `job_mgr` authorization over the entire database, then the following command will not work:

```
EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

Related Topics

- UNAUTHORIZE_SCHEDULER_USER Procedure
  The `UNAUTHORIZE_SCHEDULER_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_SCHEDULER_USER` procedure.

12.6 Using Information Lifecycle Management with Oracle Database Vault

Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.

- About Using Information Lifecycle Management with Oracle Database Vault
  You can grant authorization to and from users who are responsible for performing Information Lifecycle Management (ILM) operations on Oracle Database Vault realm- and command rule-protected objects.
- Authorizing Users for ILM Operations in Database Vault
  You can authorize a user to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

- Revoking Information Lifecycle Management Authorization from Users
  You can revoke authorization from users so that they cannot perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

12.6.1 About Using Information Lifecycle Management with Oracle Database Vault

You can grant authorization to and from users who are responsible for performing Information Lifecycle Management (ILM) operations on Oracle Database Vault realm- and command rule-protected objects.

You must first authorize users before they can perform the following SQL statements for ILM operations in a Database Vault-enabled database:

- `ALTER TABLE`
  - `ILM`
  - `FLASHBACK ARCHIVE`
  - `NO FLASHBACK ARCHIVE`
- `ALTER TABLESPACE`
  - `FLASHBACK MODE`

12.6.2 Authorizing Users for ILM Operations in Database Vault

You can authorize a user to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant the necessary authorization.

2. Use the `DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER` to authorize the user.

   For example, to grant a user authorization to perform ILM operations on the `HR.EMPLOYEES` table:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER ('PSMITH', 'HR', 'EMPLOYEES',
   'TABLE', 'ILM');
   ```

   If you wanted to grant user `psmith` ILM authorizations for the entire database, you would enter a procedure similar to the following:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER ('PSMITH', '%', '%', '%', '%');
   ```

3. Ensure that the user has been authorized by querying the `DBA_DV_MAINTENANCE_AUTH` data dictionary view.

Related Topics

- AUTHORIZE_MAINTENANCE_USER Procedure
  The `AUTHORIZE_MAINTENANCE_USER` procedure grants a user authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

- DBA_DV_MAINTENANCE_AUTH View
  The DBA_DV_MAINTENANCE_AUTH data dictionary view provides information about the configuration of Oracle Database Vault authorizations to use Information Life Management (ILM) features.

12.6.3 Revoking Information Lifecycle Management Authorization from Users

You can revoke authorization from users so that they cannot perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.

   Only a user who has been granted either of these roles can grant the necessary authorization.
2. Query the DBA_DV_MAINTENANCE_AUTH data dictionary view to find the kind of authorization that was granted to the ILM user.
3. Use the DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER to revoke the authorization from the user.

   For example:

```
EXEC DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER ('PSMITH', 'HR', '%', 'TABLE',
'ILM');
```

Related Topics

- DBA_DV_MAINTENANCE_AUTH View
  The DBA_DV_MAINTENANCE_AUTH data dictionary view provides information about the configuration of Oracle Database Vault authorizations to use Information Life Management (ILM) features.
- UNAUTHORIZE_MAINTENANCE_USER Procedure
  The UNAUTHORIZE_MAINTENANCE_USER procedure revokes privileges from users who have been granted authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

12.7 Using Oracle Database Replay with Oracle Database Vault

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

- About Using Database Replay with Oracle Database Vault
  You can grant Database Vault authorizations for users to perform both workload capture and workload replay operations with Oracle Database Replay.
- Authorizing Users for Database Replay Operations
  You can authorize Oracle Database Replay users for both workload capture and workload replay operations.
- Revoking Database Replay Authorization from Users
  You can remove authorization for both Oracle Database Replay workload capture and workload replay operations.

12.7.1 About Using Database Replay with Oracle Database Vault

You can grant Database Vault authorizations for users to perform both workload capture and workload replay operations with Oracle Database Replay.

12.7.2 Authorizing Users for Database Replay Operations

You can authorize Oracle Database Replay users for both workload capture and workload replay operations.

- Authorizing Users for Workload Capture Operations
  You can authorize a user to perform Oracle Database Replay workload capture operations in an Oracle Database Vault environment.
- Authorizing Users for Workload Replay Operations
  You can authorize a user to perform Oracle Database Replay workload replay operations in an Oracle Database Vault environment.

12.7.2.1 Authorizing Users for Workload Capture Operations

You can authorize a user to perform Oracle Database Replay workload capture operations in an Oracle Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant this authorization.

2. Use the `DBMS_MACADM.AUTHORIZE_DBCAPTURE` procedure to authorize the user.

   For example:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DBCAPTURE ('PFITCH');
   ```

3. Ensure that the user has been authorized by querying the `DBA_DV_DBCAPTURE_AUTH` data dictionary view.

Related Topics

- AUTHORIZE_DBCAPTURE Procedure
  The `AUTHORIZE_DBCAPTURE` procedure grants a user authorization to perform Oracle Database Replay workload capture operations.
- DBA_DV_DBCAPTURE_AUTH View
  The `DBA_DV_DBCAPTURE_AUTH` data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload capture operations.

12.7.2.2 Authorizing Users for Workload Replay Operations

You can authorize a user to perform Oracle Database Replay workload replay operations in an Oracle Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant this authorization.

2. Use the `DBMS_MACADM.AUTHORIZE_DBREPLAY` procedure to authorize the user.

For example:

```
EXEC DBMS_MACADM.AUTHORIZE_DBREPLAY ('PFITCH');
```

3. Ensure that the user has been authorized by querying the `DBA_DV_DBREPLAY_AUTH` data dictionary view.

Related Topics

- AUTHORIZE_DBREPLAY Procedure
  The `AUTHORIZE_DBREPLAY` procedure grants a user authorization to perform Oracle Database Replay workload replay operations.
- DBA_DV_DBREPLAY View
  The `DBA_DV_DBREPLAY_AUTH` data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload replay operations.

12.7.3 Revoking Database Replay Authorization from Users

You can remove authorization for both Oracle Database Replay workload capture and workload replay operations.

- Revoking Workload Capture Privileges
  You can revoke authorization from users so that they cannot perform Oracle Database Replay workload capture operations in an Oracle Database Vault environment.
- Revoking Workload Replay Privileges
  You can revoke authorization from users so that they cannot perform Oracle Database Replay workload replay operations in an Oracle Database Vault environment.

12.7.3.1 Revoking Workload Capture Privileges

You can revoke authorization from users so that they cannot perform Oracle Database Replay workload capture operations in an Oracle Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant this authorization.

2. Query the `DBA_DV_DBCAPTURE_AUTH` data dictionary view to find users whose workload capture authorization you want to revoke.
3. Use the `DBMS_MACADM.UNAUTHORIZE_DBCAPTURE` procedure to revoke authorization from the user.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_DBCAPTURE ('PFITCH');
   ```

Related Topics

- DBA_DV_DBCAPTURE_AUTH View
  The `DBA_DV_DBCAPTURE_AUTH` data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload capture operations.
- UNAUTHORIZE_DBCAPTURE Procedure
  The `UNAUTHORIZE_DBCAPTURE` procedure revokes authorization from users to perform Oracle Database Replay workload capture operations.

12.7.3.2 Revoking Workload Replay Privileges

You can revoke authorization from users so that they cannot perform Oracle Database Replay workload replay operations in an Oracle Database Vault environment.

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

   Only a user who has been granted either of these roles can grant this authorization.
2. Query the `DBA_DV_DBREPLAY_AUTH` data dictionary view to find users whose workload replay authorization you want to revoke.
3. Use the `DBMS_MACADM.UNAUTHORIZE_DBDBREPLAY` procedure to revoke authorization from the user.

   For example:

```
EXEC DBMS_MACADM.UNAUTHORIZE_DBREPLAY ('PFITCH');
```

Related Topics

- DBA_DV_DBREPLAY View
  The `DBA_DV_DBREPLAY_AUTH` data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload replay operations.
- UNAUTHORIZE_DBREPLAY Procedure
  The `UNAUTHORIZE_DBREPLAY` procedure revokes authorization from users to perform Oracle Database Replay workload replay operations.

12.8 Running Preprocessor Programs with Oracle Database Vault

Users who run preprocessor programs through external tables must have Oracle Database Vault-specific authorization.

- About Running Preprocessor Programs with Oracle Database Vault
  You can grant and revoke Database Vault authorizations for users to run preprocessor programs through external tables.
- Authorizing Users to Run Preprocessor Programs
  The `DBMS_MACADM.AUTHORIZE_PREPROCESSOR` procedure grants users authorization to run preprocessor programs through external tables.
- Revoking Authorization to Run Execute Preprocessor Programs from Users
  The `DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR` procedure revokes authorization from users so that they cannot run preprocessor programs through external tables in an Oracle Database Vault environment.

12.8.1 About Running Preprocessor Programs with Oracle Database Vault

You can grant and revoke Database Vault authorizations for users to run preprocessor programs through external tables.

12.8.2 Authorizing Users to Run Preprocessor Programs

The `DBMS_MACADM.AUTHORIZE_PREPROCESSOR` procedure grants users authorization to run preprocessor programs through external tables.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant this authorization.

2. Use the `DBMS_MACADM.AUTHORIZE_PREPROCESSOR` procedure to authorize the user.

   For example:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_PREPROCESSOR ('PFITCH');
   ```

3. Ensure that the user has been authorized by querying the `DBA_DV_PREPROCESSOR_AUTH` data dictionary view.

12.8.3 Revoking Authorization to Run Execute Preprocessor Programs from Users

The `DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR` procedure revokes authorization from users so that they cannot run preprocessor programs through external tables in an Oracle Database Vault environment.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.

   Only a user who has been granted either of these roles can grant this authorization.

2. Use the `DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR` procedure to revoke the authorization from the user.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR ('PFITCH');
   ```

3. Query the `DBA_DV_PREPROCESSOR_AUTH` data dictionary view to ensure that the user is no longer authorized.

12.9 Using Database Vault Operations Control to Restrict Multitenant Common User Access to Local PDB Data

You can control PDB access by CDB root common users, such as infrastructure database administrators.

- About Using Database Vault Operations Control
  You can automatically restrict common users from accessing pluggable database (PDB) local data in autonomous, regular Cloud, or on-premises environments.
- How the Addition of Common Users and Packages to an Exception List Works
  Before you add a common user or package to an exception list, they must fulfill special requirements.
- Enabling Database Vault Operations Control
  To enable Database Vault operations control, use the `DBMS_MACADM.ENABLE_APP_PROTECTION` PL/SQL procedure.

- Adding Common Users and Packages to an Exception List
  Common users and applications that must access PDB local data can be added to an exception list.
- Deleting Common Users and Packages from an Exception List
  Users and applications that no longer need to access PDB local data can be removed from the exception list.
- Disabling Database Vault Operations Control
  To disable Database Vault operations control, use the `DBMS_MACADM.DISABLE_APP_PROTECTION` PL/SQL procedure.

12.9.1 About Using Database Vault Operations Control

You can automatically restrict common users from accessing pluggable database (PDB) local data in autonomous, regular Cloud, or on-premises environments.

To accomplish this, you can use Oracle Database Vault operations control, which applies to common users such as infrastructure database administrators and applications.

Database Vault operations control is useful for situations where a database administrator must log in to the CDB root as a highly privileged user, but still not be able to access PDB customer data. Database operations control does not block PDB database administrators. To block these users, enable Oracle Database Vault in the PDB and then use the Database Vault features such as realm control to block these users.

You can create an exception list for Database Vault operations control of common users and packages for situations where a common user or application must perform tasks that must access local data on a PDB. An example of the type of common user that you would specify for the exception list is the `CTXSYS` application account, which is responsible for Oracle Text. Specifying a package in an exception list enables you to apply more fine-grained control instead of providing full access to a user in an exception list.

The general process for using Database Vault operations control is as follows:

1. Enable Database Vault operations control and keep it enabled for the production environment.
2. At this stage Database Vault operations control applies to all PDBs in the environment, regardless of whether the PDB has enabled Database Vault or not.
3. To enable specific users and packages to have access to the local schemas of the PDBs, add them to an exception list. When the user or package no longer needs access, then you can remove them from the exception list. For example, if the database is using Oracle Text, then you can add the `CTXSYS` administrative user account and the package to the exception list.

12.9.2 How the Addition of Common Users and Packages to an Exception List Works

Before you add a common user or package to an exception list, they must fulfill special requirements.

You can add a user package to the exception list if the package is the only object in the user account that needs access to the PDB local data. This allows for fine grained control over what is put into the exception list. The kinds of common users and packages that you would add to the exception list are ones that are necessary for the functioning of the PDB. For example, if you are using Oracle Spatial, then you should add the `MDSYS` account to the exception list. `MDSYS` requires access to customer PDB data for Oracle Spatial functions.

A PL/SQL procedure on the Ops Control exception list can be run by any common user, as long as the common user has system or direct object privileges to run the PL/SQL procedure. (Only definer's rights procedures can be added to the exception list, not invoker's rights.)

Only users on the operations control exception list (user, `%` exception) can modify PL/SQL procedures on an exception list and only if they have the privileges to modify the PL/SQL procedures. For example, User X cannot modify their own User X PL/SQL procedure if the procedure is on the operations control exception list, but User X is not on the exception list. User Y can modify User X procedures if User Y is on the exception list (Y, `%`) and if User Y has privileges to modify User X procedures.

To add a common user and a package to the Database Vault operations control exception list, you can use the `DBMS_MACADM.ADD_APP_EXCEPTION` PL/SQL procedure. To find existing exceptions, you can query the `DBA_DV_APP_EXCEPTION` data dictionary view.

12.9.3 Enabling Database Vault Operations Control

To enable Database Vault operations control, use the `DBMS_MACADM.ENABLE_APP_PROTECTION` PL/SQL procedure.

Oracle recommends that if you elect to use Database Vault operations control for your multitenant production server, then you should keep Database Vault operations control enabled full time.

In most cases, you will enable Database Operations control for the entire CDB, not just a specific PDB. If you need to disable it for a specific PDB (for example, for troubleshooting purposes), then you can run the `DBMS_MACADM.DISABLE_APP_PROTECTION` procedure on the PDB. When you are finished troubleshooting the PDB, re-enable it for Database Vault operations control, as shown in the example in this topic.

Before you enable Database Vault operations control, Database Vault must be enabled and configured in the CDB root. However, Database Vault does not need to be enabled in the PDBs.

1. Log in to the CDB root as a common user who has been granted the `DV_OWNER` role.
2. Run the `DBMS_MACADM.ENABLE_APP_PROTECTION` procedure.
   - To enable Database Vault operations control for all PDBs in the CDB environment:

     ```
     EXEC DBMS_MACADM.ENABLE_APP_PROTECTION;
     ```

   - The operations control for a specific PDB may have been disabled for troubleshooting reasons. To re-enable Database Vault operations control for a specific PBB (for example, `HRPDB`):

     ```
     EXEC DBMS_MACADM.ENABLE_APP_PROTECTION ('HRPDB');
     ```

At this stage, one or all of the PDBs are enabled for Database Vault operations control. You can confirm by connecting as user `SYS` with the `SYSDBA` administrative privilege and then

executing the `SELECT * FROM DBA_DV_STATUS;` query. If specific trusted common users or packages must have access to the local schemas of these PDBs to perform special operations, then you can use the `DBMS_MACADM.ADD_APP_EXCEPTION` procedure to add the user or package to an exception list for Database Vault operations control.

Related Topics

- Adding Common Users and Packages to an Exception List
  Common users and applications that must access PDB local data can be added to an exception list.

12.9.4 Adding Common Users and Packages to an Exception List

Common users and applications that must access PDB local data can be added to an exception list.

1. Log in to the CDB root as a common user who has been granted the `DV_OWNER` role.
2. Ensure that the package that you will specify for the common user meets the following requirements:
   - The package must be owned by the common user.
   - A user-created package must be created with definer's rights procedures.

   You can find more information about user-created packages by querying the `DBA_OBJECTS` data dictionary view.
3. Execute the `DBMS_MACADM.ADD_APP_EXCEPTION` procedure.

   For example:

```
DBMS_MACADM.ADD_APP_EXCEPTION ('MDSYS', 'PATCH_APP');
```

12.9.5 Deleting Common Users and Packages from an Exception List

Users and applications that no longer need to access PDB local data can be removed from the exception list.

To remove a common user and a package from the Database Vault operations control exception list, you can use the `DBMS_MACADM.DELETE_APP_PROTECTION` PL/SQL procedure. To find existing exceptions, you can query the `DBA_DV_APP_EXCEPTION` data dictionary view.

1. Log in to the CDB root as a common user who has been granted the `DV_OWNER` role.
2. Run the `DBMS_MACADM.DELETE_APP_EXCEPTION` procedure.

   For example:

```
DBMS_MACADM.DELETE_APP_EXCEPTION ('MDSYS', 'PATCH_APP');
```

12.9.6 Disabling Database Vault Operations Control

To disable Database Vault operations control, use the `DBMS_MACADM.DISABLE_APP_PROTECTION` PL/SQL procedure.

In most cases, you should keep Database Vault operations control enabled. If troubleshooting requires that a PDB be dropped from Database Vault operations control, then Oracle recommends that you temporarily disable Database Vault operations control for the PDB (and maintain operations control for the rest of the PDBs). After the troubleshooting is complete, then you should re-enable Database Vault operations control.

1. Log in to the CDB root as a common user who has been granted the `DV_OWNER` role.
2. Run the `DBMS_MACADM.DISABLE_APP_PROTECTION` procedure.
   - To disable Database Vault operations control for all PDBs in the CDB environment:

     ```
     EXEC DBMS_MACADM.DISABLE_APP_PROTECTION;
     ```

   - To disable Database Vault operations control for a specific PBB (for example, `HRPDB`):

     ```
     EXEC DBMS_MACADM.DISABLE_APP_PROTECTION ('HRPDB');
     ```

12.10 Preventing Multitenant Local Users from Blocking Common Operations

You can prevent multitenant local users from blocking common operations when they attempt to create Oracle Database Vault protections on common user objects.

- About Preventing Multitenant Local Users from Blocking Common Operations
  A user who has the `DV_OWNER` role in the root can control whether local PDB users can create Oracle Database Vault controls on a common user's local objects.
- Preventing Local Users from Blocking Common Operations
  To prevent local users from blocking common operations, run the `DBMS_MACADM.ALLOW_COMMON_OPERATION` procedure in the root.

12.10.1 About Preventing Multitenant Local Users from Blocking Common Operations

A user who has the `DV_OWNER` role in the root can control whether local PDB users can create Oracle Database Vault controls on a common user's local objects.

If a local user can apply Oracle Database Vault controls (such as realms or command rules) to a local object that is owned by a common user, or to an object owned by an application common user, then the common user or the application common user could be blocked from accessing local data in their own schema in the PDB. This may prevent them from running common operations necessary for the maintenance of the database or application. In addition, a local user could be able to create a CONNECT command rule on a common user that can prevent this common user from logging in to the PDB in which the common user's objects reside.

To prevent local users from being able to block common operations, a common user who has been granted the DV_OWNER role in the root can run the DBMS_MACADM.ALLOW_COMMON_OPERATION procedure in the root.

To find the current status of how DBMS_MACADM.ALLOW_COMMON_OPERATION has been set, a user with the DV_OWNER or DV_ADMIN role can query the DVSYS.DBA_DV_COMMON_OPERATION_STATUS data dictionary view.

Related Topics

- DVSYS.DBA_DV_COMMON_OPERATION_STATUS View
  The DVSYS.DBA_DV_COMMON_OPERATION_STATUS data dictionary view displays the status of the DBMS_MACADM.ALLOW_COMMON_OPERATION procedure setting.

12.10.2 Preventing Local Users from Blocking Common Operations

To prevent local users from blocking common operations, run the DBMS_MACADM.ALLOW_COMMON_OPERATION procedure in the root.

When you set ALLOW COMMON OPERATION to TRUE, then local users are restricted from creating Oracle Database Vault controls on common user objects. This setting applies to existing local PDB Database Vault controls that were created on common user objects, so that they will not be enforced on common users.

1. Log in to the root as a user who has been granted the DV_OWNER role for the root.
2. Run the DBMS_MACADM.ALLOW_COMMON_OPERATION procedure as follows:

   ```
   EXEC DBMS_MACADM.ALLOW_COMMON_OPERATION (TRUE);
   ```

   In this specification:

   - TRUE prevents local users from creating Oracle Database Vault controls on common user objects. Alternatively, you can run this procedure without including any parameter to achieve a TRUE result.
   - FALSE enables local users to create Database Vault controls on common user objects. If you do not run DBMS_MACADM.ALLOW_COMMON_OPERATION at all, then the default ALLOW COMMON OPERATION status is FALSE, and the default behavior will be to allow local users to create Database Vault controls on common user objects.

   If a realm or command rule was already created on a common object while DBMS_MACADM.ALLOW_COMMON_OPERATION is set to FALSE, and then subsequently, DBMS_MACADM.ALLOW_COMMON_OPERATION is set to TRUE, then the realm and command rule on the common object are not enforced.

12.11 Oracle Recovery Manager and Oracle Database Vault

You can use Recovery Manager (RMAN) in an Oracle Database Vault environment.

The functionality of RMAN with Oracle Database Vault is almost the same as its functionality in a standard Oracle Database environment. However, be aware that the RMAN recover table and table partitions features do not work with realm-protected tables when you attempt an export operation. To perform an export operation, you must perform a full table recovery and then have a Database Vault authorized user perform the export of the real-protected protected table.

Be aware that the RMAN recover table and table partitions features do not work with realm-protected tables when you attempt to recover the table. To recover the table, you must perform a full database recovery and then have a Database Vault authorized user perform the export of the realm-protected table to import into the existing database.

Related Topics

- *Oracle Database Backup and Recovery User's Guide*
- *Oracle Database Backup and Recovery Reference*

12.12 Privileges for Using XStream with Oracle Database Vault

If you want to use XStream in an Oracle Database Vault environment, then you must have the appropriate privileges.

These privileges are as follows:

- You must be granted the `DV_XSTREAM_ADMIN` role in order to configure the XStream.
- Before you can apply changes to any tables that are protected by a realm, you must be authorized to have access to that realm. For example:

  ```
  EXEC DBMS_MACADM.ADD_AUTH_TO_REALM('realm_name','username');
  ```

- Before you can run the `DBMS_XSTREAM_AUTH.GRANT_ADMIN_PRIVILEGE` procedure, you must be granted the `DV_ACCTMGR` role.

Related Topics

- DV_XSTREAM_ADMIN XStream Administrative Role
  The `DV_XSTREAM_ADMIN` role is used for Oracle XStream.
- ADD_AUTH_TO_REALM Procedure
  The `ADD_AUTH_TO_REALM` procedure authorizes a user or role to access a realm as an owner or a participant. You can authenticate both common and local realms.

12.13 Privileges for Using Oracle GoldenGate with Oracle Database Vault

If you want to use Oracle GoldenGate in an Oracle Database Vault environment, then you must have the appropriate privileges.

These privileges are as follows:

- The user must be granted the `DV_GOLDENGATE_ADMIN` role in order to configure the Oracle GoldenGate.
- The user must be granted the `DV_GOLDENGATE_REDO_ACCESS` role if the user must use the Oracle GoldenGate `TRANLOGOPTIONS DBLOGREADER` method to access redo logs.

  For example, to grant the `DV_GOLDENGATE_ADMIN` and `DV_GOLDENGATE_REDO_ACCESS` roles to a user named `gg_admin`:

  ```
  GRANT DV_GOLDENGATE_ADMIN, DV_GOLDENGATE_REDO_ACCESS TO gg_admin;
  ```

- The user must be granted the `DV_ACCTMGR` role before this user can create users on the replicated side.

- The user must perform extract operations in triggerless mode before attempting to perform procedural replication.
- Before users can apply changes to any tables that are protected by a realm, they must be authorized to have access to that realm. For example:

```
EXEC DBMS_MACADM.ADD_AUTH_TO_REALM('realm_name','username');
```

- The `SYS` user must be authorized to perform Data Definition Language (DDL) operations in the `SYSTEM` schema, as follows:

```
EXECUTE DVSYS.DBMS_MACADM.AUTHORIZE_DDL('SYS', 'SYSTEM');
```

- The user must be granted authorization to the Oracle Default Component Protection Realm. For example, to grant this realm authorization to a user named `gg_admin`:

```
BEGIN
 DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM(
   REALM_NAME    => 'Oracle Default Component Protection Realm',
   GRANTEE       => 'gg_admin',
   AUTH_OPTIONS  => 1);
END;
/
```

> **Note:**
>
> Oracle GoldenGate queries, updates, and manages objects in the `SYS`, `SYSTEM` and GoldenGate-related schemas. If any of the schemas are protected by an Oracle Database Vault realm, then the GoldenGate Extract operation can fail. Oracle Database Vault protects dictionary related objects with the `Oracle Default Component Protection Realm` and recommends that you do not protect default schemas, such as `SYS` and `SYSTEM`, with any custom Oracle Database Vault realms or custom Oracle Database Vault command rules.

Related Topics

- DV_GOLDENGATE_ADMIN GoldenGate Administrative Role
  The `DV_GOLDENGATE_ADMIN` role is used with Oracle GoldenGate.
- DV_GOLDENGATE_REDO_ACCESS GoldenGate Redo Log Role
  The `DV_GOLDENGATE_REDO_ACCESS` role is used with Oracle GoldenGate.
- ADD_AUTH_TO_REALM Procedure
  The `ADD_AUTH_TO_REALM` procedure authorizes a user or role to access a realm as an owner or a participant. You can authenticate both common and local realms.

12.14 Using Data Masking in an Oracle Database Vault Environment

You must have the correct authorization to perform data masking in an Oracle Database Vault environment.

- About Data Masking in an Oracle Database Vault Enabled Database
  In an Oracle Database Vault-enabled database, only users who have Database Vault authorizations can mask data in Database Vault-protected database objects.
- Adding Data Masking Users to the Data Dictionary Realm Authorizations
  You can add data masking users to the Oracle Default Component Protection realm to give them data dictionary realm authorizations.
- Giving Users Access to Tables or Schemas That They Want to Mask
  To give users access to tables or schemas that they want to mask, you must authorize them for the appropriate realm.
- Creating a Command Rule to Control Data Masking Privileges
  You must have privileges to manage tables, packages, and triggers before you can use data masking in an Oracle Database Vault environment.

12.14.1 About Data Masking in an Oracle Database Vault Enabled Database

In an Oracle Database Vault-enabled database, only users who have Database Vault authorizations can mask data in Database Vault-protected database objects.

In a non-Database Vault environment, users who have been granted the `SELECT_CATALOG_ROLE` and `DBA` roles can perform data masking. However, with Database Vault, users must have additional privileges. This section describes three ways that you can use to enable users to mask data in Database Vault-protected objects.

If users do not have the correct privileges, then the following errors can occur while creating the masking definition or when the job is executing:

```
ORA-47400: Command Rule violation for string on string

ORA-47401: Realm violation for string on string.

ORA-47408: Realm violation for the EXECUTE command

ORA-47409: Command Rule violation for the EXECUTE command

ORA-01301: insufficient privileges
```

12.14.2 Adding Data Masking Users to the Data Dictionary Realm Authorizations

You can add data masking users to the Oracle Default Component Protection realm to give them data dictionary realm authorizations.

The Oracle Data Dictionary controls access to the Oracle Database catalog schemas, such as `SYS` and `SYSTEM`. (See Default Realms for a full list of these schemas.) It also controls the ability to grant system privileges and database administrator roles. If you add users to the Oracle Default Component Protection realm, and assuming these users already have the privileges associated with the Oracle Data Dictionary, then these users will have these same privileges in a Database Vault environment. Therefore, if you do add a user to this realm, ensure that this user is a trusted user.

- To add a user to the Oracle Default Component Protection realm, use the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure.

For example:

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name   => 'Oracle Default Component Protection Realm',
  grantee      => 'DBA_JSMITH',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT);
END;
/
```

12.14.3 Giving Users Access to Tables or Schemas That They Want to Mask

To give users access to tables or schemas that they want to mask, you must authorize them for the appropriate realm.

If the table or schema of a table that is to be data masked is in a realm, then you must add the user responsible for data masking to the realm authorization as a participant or owner. If the table or schema has dependent objects that are in other realm-protected tables, then you must grant the user participant or owner authorization for those realms as well.

- To authorize users for data masking to a realm that protects the objects they want to data mask, use the `DBMS_MACADM.ADD_AUTH_TO_REALM` procedure.

The following example shows how to grant user `DBA_JSMITH` authorization for the `HR.EMPLOYEES` table, which is protected by a realm called Business Apps Realm:

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name   => 'Business Apps Realm',
  grantee      => 'DBA_JSMITH',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT;
END;
/
```

12.14.4 Creating a Command Rule to Control Data Masking Privileges

You must have privileges to manage tables, packages, and triggers before you can use data masking in an Oracle Database Vault environment.

For data masking, users must have the `CREATE TABLE`, `SELECT TABLE`, `ALTER TABLE`, and `DROP TABLE` privileges for the masking objects and if there are any dependent objects to be created, the user must have the appropriate privileges such as `CREATE PACKAGE`, `CREATE TRIGGER`, and so on.

You can create command rules to control data masking privileges at a granular level. To do so, create a command rule that can either prevent or allow the user access to objects that must have to be data masked. For example, you can create a command rule called Allow Data Masking that checks if the user is in a list of users who are responsible for data masking. If the user logging in is one of these users, then the command rule evaluates to true and the user is permitted to create the data mask for the protected object.

To create a command rule that controls data masking privileges:

1. Create the rule set rule.

For example:

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Is HDRISCOLL or DBA_JSMITH User',
  rule_expr  =>'USER IN(''HDRISCOLL'',''DBA_JSMITH'')');
END;
/
```

2. Create a rule set and then add the rule to it:

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Allow Data Masking',
  description      => 'Allows users HDRISCOLL and DBA_JSMITH access',
  enabled          => 'DBMS_MACUTL.G_YES',
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'You do not have access to this object.',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
  is_static        => TRUE);
END;
/
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Allow Data Masking',
  rule_name     => 'Is HDRISCOLL or DBA_JSMITH User'),
  rule_order    => 1);
END;
/
```

3. Create a command rule and then add this rule to it:

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'CREATE TABLE',
  rule_set_name   => 'Allow Data Masking',
  object_owner    => 'HR',
  object_name     => 'EMPLOYEES',
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

12.15 Using Oracle Database Auditing with Oracle Database Vault

You grant and revoke authorizations to manage and view audit records in an Oracle Database Vault environment.

- About Using Oracle Database Auditing with Oracle Database Vault
  Controlling who has the authority to grant and revoke auditing authorizations in an Oracle Database Vault environment ensures a tighter control of audit management.
- Authorizing the Management of Audit Policies and Related Auditing Tasks
  You can authorize a user to perform tasks that are associated with the Oracle Database `AUDIT_ADMIN` role.

- Authorizing the Ability to View and Analyze Audit Data
  You can authorize a user to perform tasks that are associated with the Oracle Database `AUDIT_VIEWER` role.

12.15.1 About Using Oracle Database Auditing with Oracle Database Vault

Controlling who has the authority to grant and revoke auditing authorizations in an Oracle Database Vault environment ensures a tighter control of audit management.

> **Note:**
>
> Starting with Oracle Database release 23c, traditional auditing is desupported.

For users who are responsible for auditing tasks such as creating and managing unified audit policies, you can grant and revoke the authorizations that are provided by the Oracle Database `AUDIT_ADMIN` and `AUDIT_VIEWER` roles. Note that these roles are specific to Oracle Database and are not part of Oracle Database Vault.

- **AUDIT_ADMIN role.** This Oracle Database role enables you to create unified and fine-grained audit policies, use the `AUDIT` and `NOAUDIT` SQL statements, view audit data, and manage the audit trail administration. Only trusted users should be granted this role.
- **AUDIT_VIEWER role.** This Oracle Database role enables users to view and analyze audit data. It provides the `EXECUTE` privilege on the `DBMS_AUDIT_UTIL` PL/SQL package. The kind of user who needs this role is typically an external auditor.

Oracle Database Vault protects the `AUDSYS` schema and audit-related objects in the `SYS` schema (for example, audit-related `SYS`-owned tables, views, and packages) with the `Oracle Audit` realm, which is a mandatory realm. In addition, this authorization prevents direct DMLs and DDLs on the `SYS.AUD$` and `SYS.FGA_LOG$` tables.

Related Topics

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.
- Oracle Audit Realm
  The `Oracle Audit` realm protects the `AUDSYS` schema and audit-related objects in the `SYS` schema.

12.15.2 Authorizing the Management of Audit Policies and Related Auditing Tasks

You can authorize a user to perform tasks that are associated with the Oracle Database `AUDIT_ADMIN` role.

- Granting AUDIT_ADMIN Authorization
  You can authorize a database administrator to perform audit-related tasks that are enabled for users who have been granted the Oracle Database `AUDIT_ADMIN` role.
- Revoking AUDIT_ADMIN Authorization
  You can revoke authorization from the database administrator who has been granted `AUDIT_ADMIN` authorization.

12.15.2.1 Granting AUDIT_ADMIN Authorization

You can authorize a database administrator to perform audit-related tasks that are enabled for users who have been granted the Oracle Database `AUDIT_ADMIN` role.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Ensure that the user to whom you want to grant the `AUDIT_ADMIN` authorization has been granted the Oracle Database `AUDIT_ADMIN` role.

   For example:

   ```
   SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE =
   'AUDIT_ADMIN';
   ```

3. Grant the user `AUDIT_ADMIN` authorization.

   For example:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN ('PSMITH');
   ```

Related Topics

- AUTHORIZE_AUDIT_ADMIN Procedure
  The `AUTHORIZE_AUDIT_ADMIN` procedure enables a user to perform the auditing tasks that are available to users who have been granted the Oracle Database `AUDIT_ADMIN` role, such as creating audit policies.
- DBA_DV_AUDIT_ADMIN_AUTH View
  The `DBA_DV_AUDIT_ADMIN_AUTH` data dictionary view lists users who have `AUDIT_ADMIN` authorization.

12.15.2.2 Revoking AUDIT_ADMIN Authorization

You can revoke authorization from the database administrator who has been granted `AUDIT_ADMIN` authorization.

1. Log into the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Query the `DBA_DV_AUDIT_ADMIN_AUTH` to check if the user currently has `AUDIT_ADMIN` authorization.

   For example:

   ```
   SELECT GRANTEE FROM DBA_DV_AUDIT_ADMIN_AUTH;
   ```

3. Revoke the `AUDIT_ADMIN` authorization.

For example:

```
EXEC DBMS_MACADM.UNAUTHORIZE_AUDIT_ADMIN (PSMITH);
```

4. If necessary, revoke the Oracle Database AUDIT_ADMIN role from this user.

   For example:

   ```
   REVOKE AUDIT_ADMIN FROM PSMITH;
   ```

Related Topics

- UNAUTHORIZE_ADMIN_USER Procedure
  The UNAUTHORIZE_AUDIT_ADMIN procedure revokes the AUDIT_ADMIN authorization from a user.
- DBA_DV_AUDIT_ADMIN_AUTH View
  The DBA_DV_AUDIT_ADMIN_AUTH data dictionary view lists users who have AUDIT_ADMIN authorization.

12.15.3 Authorizing the Ability to View and Analyze Audit Data

You can authorize a user to perform tasks that are associated with the Oracle Database AUDIT_VIEWER role.

- Granting AUDIT_VIEWER Authorization
  You can authorize a database administrator to perform audit-related tasks that are enabled for users who have been granted the Oracle Database AUDIT_VIEWER role.
- Revoking AUDIT_VIEWER Authorization
  You can revoke authorization from the database administrator who has been granted AUDIT_VIEWER authorization.

12.15.3.1 Granting AUDIT_VIEWER Authorization

You can authorize a database administrator to perform audit-related tasks that are enabled for users who have been granted the Oracle Database AUDIT_VIEWER role.

1. Log into the PDB as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. Ensure that the user to whom you want to grant the AUDIT_VIEWER authorization has been granted the Oracle Database AUDIT_VIEWER role.

   For example:

   ```
   SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE
   = 'AUDIT_VIEWER';
   ```

3. Grant the user AUDIT_VIEWER authorization.

   For example:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_AUDIT_VIEWER ('RLAYTON');
   ```

Related Topics

- AUTHORIZE_AUDIT_VIEWER Procedure
  The AUTHORIZE_AUDIT_VIEWER procedure enables a user to perform the auditing tasks that are available to users who have been granted the AUDIT_VIEWER role to query audit-related views.
- DBA_DV_AUDIT_VIEWER_AUTH View
  The DBA_DV_AUDIT_VIEWER_AUTH data dictionary view lists users who have AUDIT_VIEWER authorization.

12.15.3.2 Revoking AUDIT_VIEWER Authorization

You can revoke authorization from the database administrator who has been granted AUDIT_VIEWER authorization.

1. Log into the PDB as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. Query the DBA_DV_AUDIT_VIEWER_AUTH to check if the user currently has AUDIT_VIEWER authorization.

   For example:

   ```
   SELECT GRANTEE FROM DBA_DV_AUDIT_VIEWER_AUTH;
   ```

3. Revoke the AUDIT_VIEWER authorization.

   For example:

   ```
   EXEC DBMS_MACADM.UNAUTHORIZE_AUDIT_VIEWER ('RLAYTON');
   ```

4. If necessary, revoke the Oracle Database AUDIT_VIEWER role from this user.

   For example:

   ```
   REVOKE AUDIT_VIEWER FROM RLAYTON;
   ```

Related Topics

- UNAUTHORIZE_AUDIT_VIEWER Procedure
  The UNAUTHORIZE_AUDIT_VIEWER procedure revokes the AUDIT_VIEWER authorization from a user.
- DBA_DV_AUDIT_VIEWER_AUTH View
  The DBA_DV_AUDIT_VIEWER_AUTH data dictionary view lists users who have AUDIT_VIEWER authorization.

12.16 Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB

You can convert a standalone Oracle Database database from release 12c through 19c to a PDB, and then plug this PDB into a CDB.

1. Connect to the root as a user who has been granted the DV_OWNER role.

2. Grant the `DV_PATCH_ADMIN` role to user `SYS` with `CONTAINER = CURRENT`.

```
GRANT DV_PATCH_ADMIN TO SYS CONTAINER = CURRENT;
```

3. In the root, connect as user `SYS` with the `SYSOPER` system privilege.
4. Restart the database in read-only mode.

   For example:

```
SHUTDOWN IMMEDIATE
STARTUP MOUNT
ALTER DATABASE OPEN READ ONLY
```

5. Connect to the Database Vault-enabled PDB as a user who has the `DV_OWNER` role.
6. Grant the `DV_PATCH_ADMIN` role to user `SYS` in this PDB.

```
GRANT DV_PATCH_ADMIN TO SYS;
```

7. Optionally, run the `DBMS_PDB.CHECK_PLUG_COMPATIBILITY` function to determine whether the unplugged PDB is compatible with the CDB.

   When you run the function, set the following parameters:

   - `pdb_descr_file`: Set this parameter to the full path to the XML file that will contain a description of the PDB.
   - `store_report`: Set this parameter to indicate whether you want to generate a report if the PDB is not compatible with the CDB. Set it to `TRUE` to generate a report or `FALSE` to not generate a report. A generated report is stored in the `PDB_PLUG_IN_VIOLATIONS` temporary table and is generated only if the PDB is not compatible with the CDB.

   For example, to determine whether a PDB described by the `/disk1/usr/dv_db_pdb.xml` file is compatible with the current CDB, run the following PL/SQL block:

```
SET SERVEROUTPUT ON
DECLARE
  compatible CONSTANT VARCHAR2(3) :=
    CASE DBMS_PDB.CHECK_PLUG_COMPATIBILITY(
           pdb_descr_file => '/disk1/usr/dv_db_pdb.xml',
           store_report   => TRUE)
    WHEN TRUE THEN 'YES'
    ELSE 'NO'
END;
BEGIN
  DBMS_OUTPUT.PUT_LINE(compatible);
END;
/
```

   If the output is `YES`, then the PDB is compatible, and you can continue with the next step.

   If the output is `NO`, then the PDB is not compatible. You can check the `PDB_PLUG_IN_VIOLATIONS` temporary table to see why it is not compatible.

8. Create an XML file that describes the PDB.

   For example:

```
BEGIN
  DBMS_PDB.DESCRIBE(
    pdb_descr_file => '/disk1/oracle/dv_db.xml');
END;
/
```

9. Run the `CREATE PLUGGABLE DATABASE` statement, and specify the XML file in the `USING` clause. Specify other clauses when they are required.

   For example:

```
CREATE PLUGGABLE DATABASE dv_db_pdb AS CLONE USING 'dv_db.xml' NOCOPY;
```

10. Connect to the PDB that you just created as user `SYS` with the `SYSDBA` administrative privilege.
11. Run the `noncdb_to_pdb.sql` script.

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

12. Open this PDB in a read/write restricted mode.

```
ALTER PLUGGABLE DATABASE pdb_name OPEN READ WRITE RESTRICTED;
```

13. Run the following procedure to synchronize the PDB:

```
EXECUTE DBMS_PDB.SYNC_PDB;
```

14. Connect to the root as a user who has been granted the `DV_OWNER` role.
15. Revoke the `DV_PATCH_ADMIN` role from user `SYS` with `CONTAINER = CURRENT`.

```
REVOKE DV_PATCH_ADMIN FROM SYS CONTAINER = CURRENT;
```

16. Connect to the legacy Database Vault-enabled PDB as user `SYS` with the `SYSOPER` system privilege.
17. Close and then reopen the PDB.

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

18. Revoke the `DV_PATCH_ADMIN` role from user `SYS`.

```
REVOKE DV_PATCH_ADMIN FROM SYS;
```

12.17 Using the ORADEBUG Utility with Oracle Database Vault

The ORADEBUG utility is used primarily by Oracle Support to diagnose problems that may arise with an Oracle database.

You can control whether users can run the ORADEBUG utility in an Oracle Database Vault-enabled environment. In a unified auditing environment, ORADBUG commands are mandatorily audited. This control does not apply to a privileged OS user, which is the OS user with the same OS user ID as the Oracle server process. This exception is made because such a user can completely control and examine the Oracle process using other means (for example, with a debugger).

1. Log into the database instance as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. If necessary, find out if ORADEBUG is already disabled or enabled.

   ```
   SELECT * FROM DBA_DV_ORADEBUG;
   ```

3. Run one of the following procedures:
   - To disable the use of ORADEBUG:

     ```
     EXEC DBMS_MACADM.DISABLE_ORADEBUG;
     ```

   - To enable the use of ORADEBUG:

     ```
     EXEC DBMS_MACADM.ENABLE_ORADEBUG;
     ```

Related Topics

- DBA_DV_ORADEBUG View
  The DBA_DV_ORADEBUG data dictionary view indicates whether users can use the ORADEBUG utility in an Oracle Database Vault environment.
- DISABLE_ORADEBUG Procedure
  The DISABLE_ORADEBUG procedure disables the use of the ORADEBUG utility in an Oracle Database Vault environment.
- ENABLE_ORADEBUG Procedure
  The ENABLE_ORADEBUG procedure enables the use of the ORADEBUG utility in an Oracle Database Vault environment.

12.18 Performing Patch Operations in an Oracle Database Vault Environment

User SYS must have the DV_PATCH_ADMIN role to perform a patch operations on an Oracle Database Vault-enabled database.

Users who have been granted the DV_PATCH_ADMIN can also view data.

1. Connect to the CDB or the application root as a user who has been granted the DV_OWNER or DV_ADMIN role.
2. Temporarily grant the SYS user the DV_PATCH_ADMIN role.

```
GRANT DV_PATCH_ADMIN TO SYS CONTAINER=ALL;
```

If you are applying a patch to a single PDB, then you do not need to grant `DV_PATCH_ADMIN` to `SYS` on all containers.

3. After the `SYS` user has performed the patch operation, carefully following the instructions in the patch readme file, then revoke `DV_PATCH_ADMIN` from user `SYS`.

```
REVOKE DV_PATCH_ADMIN FROM SYS CONTAINER=ALL;
```

13

Oracle Database Vault Schemas, Roles, and Accounts

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

- Oracle Database Vault Schemas
  The Oracle Database Vault schemas, `DVSYS` and `DVF`, support the administration and run-time processing of Oracle Database Vault.
- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.
- Oracle Database Vault Accounts Created During Registration
  The accounts that you create during registration enable Oracle Database Vault to adhere to separation of duty concepts and provide flexibility for users based on the tasks they perform.
- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the `DV_OWNER` and `DV_ACCTMGR` roles.

13.1 Oracle Database Vault Schemas

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

- DVSYS Schema
  The `DVSYS` schema contains Oracle Database Vault database objects.
- DVF Schema
  The `DVF` schema is the owner of the Oracle Database Vault `DBMS_MACSEC_FUNCTION` PL/SQL package.

13.1.1 DVSYS Schema

The `DVSYS` schema contains Oracle Database Vault database objects.

These objects store Oracle Database Vault configuration information and support the administration and run-time processing of Oracle Database Vault.

In a default installation, the `DVSYS` schema is locked. The `DVSYS` schema also owns the `AUDIT_TRAIL$` table, which is used in traditional auditing. Traditional auditing is desupported starting in Oracle 23c. The `AUDIT_TRAIL$` table is available for databases that have been upgraded to release 23c and traditional audit policies that exist from the upgrade.

The `DVSYS` schema is considered a common schema, which means that the objects within `DVSYS` (tables, views, PL/SQL packages, and so on) are automatically available to any child

pluggable databases (PDBs). In addition, the DVSYS schema account cannot switch to other containers using the ALTER SESSION statement.

Oracle Database Vault secures the DVSYS schema by using a protected schema design. A protected schema design guards the schema against improper use of system privileges (for example, SELECT ANY TABLE, CREATE ANY VIEW, or DROP ANY).

Oracle Database Vault protects and secures the DVSYS schema in the following ways:

- The DVSYS protected schema and its administrative roles cannot be dropped. By default, the DVSYS account is locked.
- By default, users cannot directly log into the DVSYS account. To control the ability of users to directly log into this account, you can run the DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS procedure to prevent users from logging in and the DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS procedure to allow users to log in.
- Statements such as CREATE USER, ALTER USER, DROP USER, CREATE PROFILE, ALTER PROFILE, and DROP PROFILE can only be issued by a user with the DV_ACCTMGR role. A user logged in with the SYSDBA administrative privilege can issue these statements only if it is allowed to do so by modifying the Can Maintain Accounts/ Profiles rule set.
- The powerful ANY system privileges for database definition language (DDL) and data manipulation language (DML) commands are blocked in the protected schema. This means that the objects in the DVSYS schema must be created by the schema account itself. Also, access to the schema objects must be authorized through object privilege grants.
- Object privileges in the DVSYS schema can only be granted to Database Vault administrative roles in the schema. This means that users can access the protected schema only through predefined administrative roles.
- Only the protected schema account DVSYS can issue ALTER ROLE statements on Database Vault predefined administrative roles of the schema.
- 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.

Related Topics

- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.

13.1.2 DVF Schema

The DVF schema is the owner of the Oracle Database Vault DBMS_MACSEC_FUNCTION PL/SQL package.

This package contains the functions that retrieve factor identities. After you install Oracle Database Vault, the installation process locks the DVF account to better secure it. When you create a new factor, Oracle Database Vault creates a new retrieval function for the factor and saves it in this schema.

The DVF user cannot switch to other containers using the ALTER SESSION statement.

By default, users cannot directly log into the DVF account. To control the ability of users to directly log into this account, you can run the DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS procedure to prevent users from logging in and the DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS procedure to allow users to log in.

13.2 Oracle Database Vault Roles

Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.

- About Oracle Database Vault Roles
  Oracle Database Vault provides a set of roles that are required for managing Oracle Database Vault.
- Privileges of Oracle Database Vault Roles
  The Oracle Database Vault roles are designed to provide the maximum benefits of separation of duty.
- Granting Oracle Database Vault Roles to Users
  You can use Enterprise Manager Cloud Control to grant Oracle Database Vault roles to users.
- DV_ACCTMGR Database Vault Account Manager Role
  The DV_ACCTMGR role is a powerful role, used for accounts management.
- DV_ADMIN Database Vault Configuration Administrator Role
  The DV_ADMIN role controls the Oracle Database Vault PL/SQL packages.
- DV_AUDIT_CLEANUP Audit Trail Cleanup Role
  The DV_AUDIT_CLEANUP role is used for purge operations.
- DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role
  The DV_DATAPUMP_NETWORK_LINK role is used for Data Pump import operations.
- DV_GOLDENGATE_ADMIN GoldenGate Administrative Role
  The DV_GOLDENGATE_ADMIN role is used with Oracle GoldenGate.
- DV_GOLDENGATE_REDO_ACCESS GoldenGate Redo Log Role
  The DV_GOLDENGATE_REDO_ACCESS role is used with Oracle GoldenGate.
- DV_MONITOR Database Vault Monitoring Role
  The DV_MONITOR role is used for monitoring Oracle Database Vault.
- DV_OWNER Database Vault Owner Role
  The DV_OWNER role enables you to manage the Oracle Database Vault roles and its configuration.

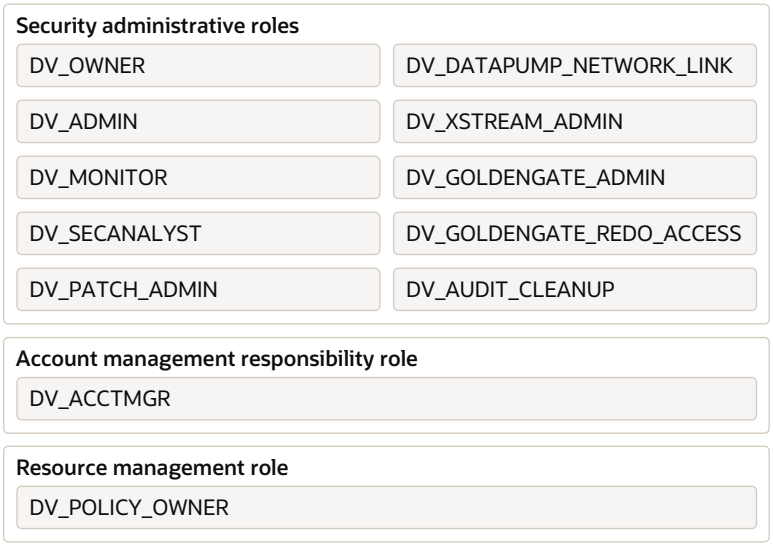
- DV_PATCH_ADMIN Database Vault Database Patch Role
  The `DV_PATCH_ADMIN` role is used for patching operations.
- DV_POLICY_OWNER Database Vault Owner Role
  The `DV_POLICY_OWNER` role enables database users to manage to a limited degree Oracle Database Vault policies.
- DV_SECANALYST Database Vault Security Analyst Role
  The `DV_SECANALYST` role enables users to analyze activities.
- DV_XSTREAM_ADMIN XStream Administrative Role
  The `DV_XSTREAM_ADMIN` role is used for Oracle XStream.

13.2.1 About Oracle Database Vault Roles

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

The following illustration shows how these roles are designed to implement the first level of separation of duties within the database. How you use these roles depends on the requirements that your company has in place.

Figure 13-1 How Oracle Database Vault Roles Are Categorized



Note:

You can grant additional object privileges and roles to the Oracle Database Vault roles to extend their scope of privileges. For example, a user logged in with the `SYSDBA` administrative privilege can grant object privileges to an Oracle Database Vault role as long as the object is not in the `DVSYS` schema or realm.

Related Topics

- Separation of Duty Guidelines
  Oracle Database Vault is designed to easily implement separation of duty guidelines.
- Managing Oracle Database Administrative Accounts
  Oracle provides guidelines for managing security for administrative accounts such as `SYSTEM` or users who have the `SYSDBA` administrative privilege.

13.2.2 Privileges of Oracle Database Vault Roles

The Oracle Database Vault roles are designed to provide the maximum benefits of separation of duty.

The `DV_PATCH_ADMIN`, `DV_XSTREAM`, `DV_GOLDENGATE_ADMIN`, and `DV_GOLDENGATE_REDO_ACCESS` roles are not included in the following sections because they have no system privileges.

DVSYS Schema, EXECUTE Privilege

Roles that can use this privilege:

- `DV_ADMIN` (includes the `EXECUTE` privilege on all Oracle Database Vault PL/SQL packages)
- `DV_OWNER` (includes the `EXECUTE` privilege on all Oracle Database Vault PL/SQL packages)
- `DV_POLICY_OWNER` (on some `DBMS_MACADM` procedures)

Roles that are denied this privilege:

- `DV_ACCTMGR`
- `DV_AUDIT_CLEANUP`
- `DV_MONITOR`
- `DV_SECANALYST`

DVSYS Schema, SELECT Privilege

Roles that can use this privilege:

- `DV_ADMIN`
- `DV_AUDIT_CLEANUP` (on some Database Vault tables and views; can perform `SELECT` statements on the `AUDIT_TRAIL$` table, and the `DV$ENFORCEMENT_AUDIT` and `DV$CONFIGURATION_AUDIT` views)
- `DV_MONITOR`
- `DV_OWNER`
- `DV_POLICY_OWNER` (on some `DBMS_MACADM` procedures and on `POLICY_OWNER*` views only)
- `DV_SECANALYST` (on some Database Vault views: `DV_SECANALYST` can query `DVSYS` schema objects through Oracle Database Vault-supplied views)

Roles that are denied this privilege:

- `DV_ACCTMGR`

DVSYS Schema, DELETE Privilege

Roles that can use this privilege:

- DV_AUDIT_CLEANUP (can perform DELETE on some Database Vault tables and views, on the AUDIT_TRAIL$ table, and the DV$ENFORCEMENT_AUDIT and DV$CONFIGURATION_AUDIT views)
- DV_OWNER (can perform DELETE on some Database Vault tables and views, on the AUDIT_TRAIL$ table, and the DV$ENFORCEMENT_AUDIT and DV$CONFIGURATION_AUDIT views)

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_ADMIN
- DV_MONITOR
- DV_POLICY_OWNER
- DV_SECANALYST

DVSYS Schema, Grant Privileges on Objects

Roles that can use this privilege: None

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_ADMIN
- DV_AUDIT_CLEANUP
- DV_MONITOR
- DV_OWNER
- DV_POLICY_OWNER
- DV_SECANALYST

DVF Schema, EXECUTE Privilege

Roles that can use this privilege:

- DV_OWNER

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_ADMIN
- DV_AUDIT_CLEANUP
- DV_MONITOR
- DV_OWNER
- DV_POLICY_OWNER
- DV_SECANALYST

DVF Schema, SELECT Privilege

Roles that can use this privilege:

- DV_OWNER
- DV_SECANALYST

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_ADMIN
- DV_AUDIT_CLEANUP
- DV_MONITOR
- DV_POLICY_OWNER

Monitor Database Vault Privilege

Roles that can use this privilege:

- DV_ADMIN
- DV_OWNER
- DV_MONITOR
- DV_SECANALYST

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_AUDIT_CLEANUP
- DV_POLICY_OWNER

Run Database Vault Reports Privilege

Roles that can use this privilege:

- DV_ADMIN
- DV_OWNER
- DV_SECANALYST

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_AUDIT_CLEANUP
- DV_MONITOR
- DV_POLICY_OWNER

SYS Schema, SELECT Privilege

Roles that can use this privilege:

- DV_MONITOR
- DV_OWNER
- DV_SECANALYST (on the same system views as DV_OWNER and DV_ADMIN)

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_ADMIN
- DV_AUDIT_CLEANUP
- DV_POLICY_OWNER

SYSMAN Schema, SELECT Privilege

Roles that can use this privilege:

- DV_OWNER (portions of SYSMAN)
- DV_SECANALYST (portions of SYSMAN)

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_ADMIN
- DV_AUDIT_CLEANUP
- DV_MONITOR
- DV_POLICY_OWNER

CREATE , ALTER , DROP User Accounts and Profiles Privilege

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

Role that can use this privilege:

- DV_ACCTMGR

Roles that are denied this privilege:

- DV_ADMIN
- DV_AUDIT_CLEANUP
- DV_MONITOR
- DV_OWNER
- DV_POLICY_OWNER
- DV_SECANALYST

Manage Objects in Schemas that Define a Realm

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

Roles that can use this privilege: None

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_AUDIT_CLEANUP
- DV_ADMIN
- DV_MONITOR

- DV_OWNER (portions of SYSMAN)
- DV_POLICY_OWNER
- DV_SECANALYST (portions of SYSMAN)

RESOURCE Role Privileges

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

Roles that can use this privilege: None

Roles that are denied this privilege:

- DV_ACCTMGR
- DV_ADMIN
- DV_AUDIT_CLEANUP
- DV_MONITOR
- DV_OWNER (portions of SYSMAN)
- DV_POLICY_OWNER
- DV_SECANALYST (portions of SYSMAN)

13.2.3 Granting Oracle Database Vault Roles to Users

You can use Enterprise Manager Cloud Control to grant Oracle Database Vault roles to users.

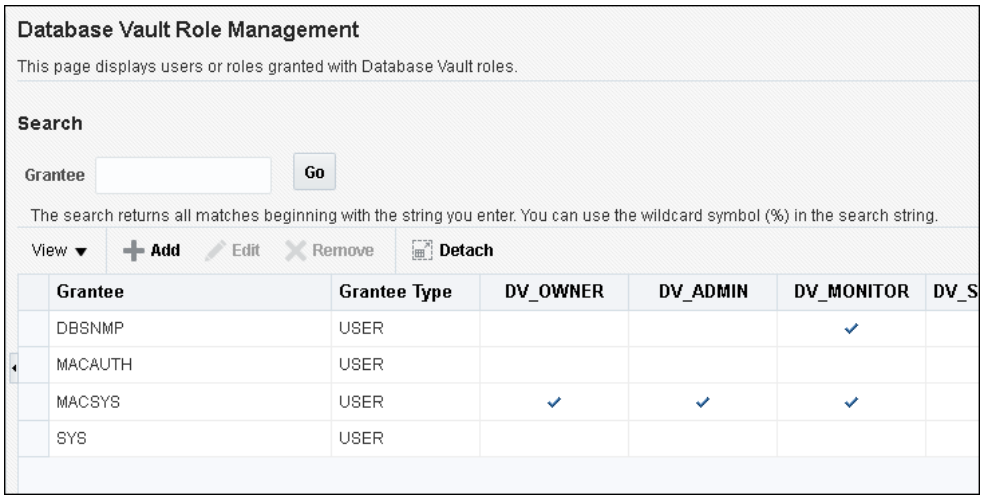
1. From Cloud Control, log into Oracle Database Vault Administrator as a user who has been granted the DV_OWNER role and the SELECT ANY DICTIONARY privilege..

   Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.

   Refer to the role descriptions to find the requirements for who can grant roles to other users.

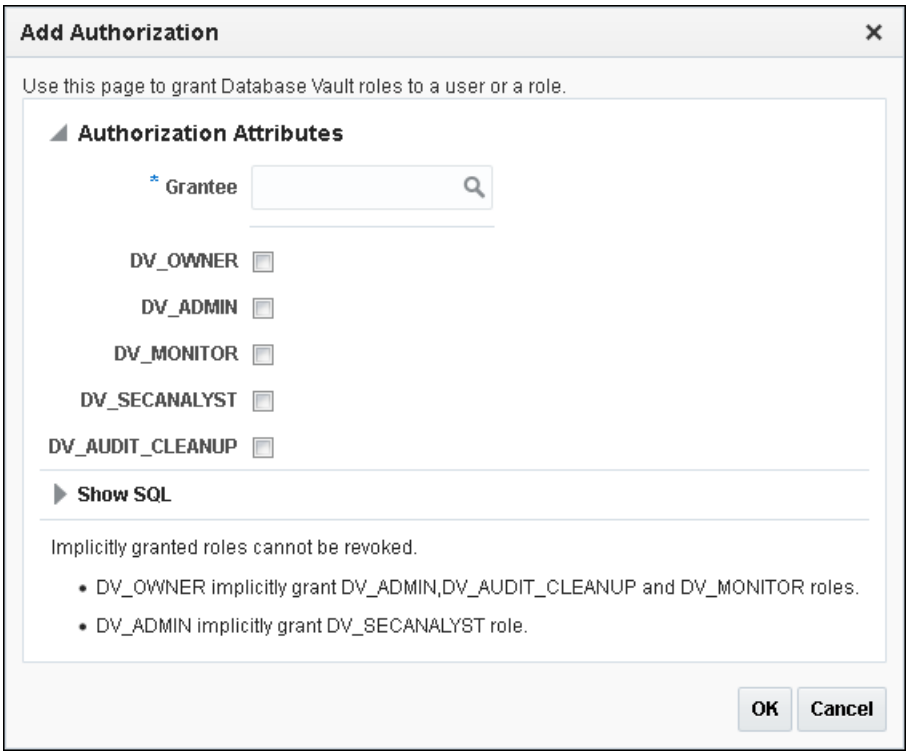
2. In the Administration page, under Database Vault Components, click **Database Vault Role Management**.

   The Database Vault Role Management page appears.



3. Do one of the following:

- To add a new user or role for a grant, click the Add button to display the Add Authorization dialog box. Enter the grantee in the **Grantee** field, and then select the roles for the grant. Then click **OK**.



- To grant different roles or modify role grants for a user or role listed in the Database Vault Role Management page, select the user or role, click **Edit**, and then modify the role grants as necessary. Then click **OK**.

13.2.4 DV_ACCTMGR Database Vault Account Manager Role

The `DV_ACCTMGR` role is a powerful role, used for accounts management.

Use the DV_ACCTMGR role to create and maintain database accounts and database profiles. In this guide, the example DV_ACCTMGR role is assigned to a user named accts_admin_acer.

Privileges Associated with the DV_ACCTMGR Role

A user who has been granted this role can use the CREATE, ALTER, and DROP statements for user accounts or profiles, including users who have been granted the DV_SECANALYST, DV_AUDIT_CLEANUP, and DV_MONITOR roles.

This user also can grant the CREATE SESSION privilege to other users. However, a person who has been granted the DV_ACCTMGR role cannot perform the following operations:

- ALTER or DROP statements on the DVSYS account
- ALTER or DROP statements on users who have been granted the DV_ADMIN or DV_OWNER role
- Change passwords for users who have been granted the DV_ADMIN or DV_OWNER role

A common user who has been granted the DV_ACCTMGR role in the CDB root can alter a common user or a common profile in the CDB root even if the common DV_ACCTMGR user does not have the SET CONTAINER privilege or the DV_ACCTMGR role in any PDB.

To find the full list of system and object privileges associated with the DV_ACCTMGR role, log into the database instance with sufficient privileges and then enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_ACCTMGR';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_ACCTMGR';
```

Tips:

- If you want the DV_ACCTMGR user to be able to grant or revoke the ANY privileges for other users, then log in as user SYS with the SYSDBA privilege and grant this user the GRANT ANY PRIVILEGE and REVOKE ANY PRIVILEGE privileges. Then add this user to the Oracle System Privilege and Role Management Realm as an owner.
- Oracle strongly recommends that you create a separate, named account for the DV_ACCTMGR user. This way, if this user forgets their password, you can log in as the original DV_ACCTMGR account and reset the user's password. Otherwise, you must disable Oracle Database Vault, log in as SYS or SYSTEM to recreate the password, and then re-enable Database Vault.

How Are GRANT and REVOKE Operations Affected by DV_ACCTMGR?

Any account, such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone does not have the rights to grant this role to or revoke this role from any other database account.

The account with the DV_ACCTMGR role and the ADMIN OPTION can grant this role to any given database account and revoke this role from another account.

DV_ACCTMGR Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.5 DV_ADMIN Database Vault Configuration Administrator Role

The DV_ADMIN role controls the Oracle Database Vault PL/SQL packages.

These packages are the underlying interface for the Database Vault Administrator user interface in Oracle Enterprise Manager Cloud Control.

Privileges Associated with the DV_ADMIN Role

The DV_ADMIN role has the EXECUTE privilege on the DVSYS packages (DBMS_MACADM and DBMS_MACUTL).

DV_ADMIN also has the capabilities provided by the DV_SECANALYST role, which allow the user to run Oracle Database Vault reports and monitor Oracle Database Vault. During installation, the DV_ADMIN role is granted to the DV_OWNER role with the ADMIN OPTION.

In addition, the DV_ADMIN role provides the SELECT privilege on the DBA_DV_POLICY, DBA_DV_POLICY_OWNER, and DBA_DV_POLICY_OBJECT data dictionary views.

To find the full list of system and object privileges associated with the DV_ADMIN role, log into the database instance with sufficient privileges and then enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE =
'DV_ADMIN';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_ADMIN';
```

How Are GRANT and REVOKE Operations Affected by DV_ADMIN?

Accounts such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone do not have the rights to grant or revoke DV_ADMIN from any other database account.

The user with the DV_OWNER role can grant or revoke this role to and from any database account.

Managing Password Changes for Users Who Have the DV_ADMIN Role

Before you can change the password for a user who has been granted the DV_ADMIN role, you must revoke the DV_ADMIN role from this account.

If you have been granted the DV_ADMIN role, then you can change your own password without having to revoke the role from yourself.

To change the DV_ADMIN user password:

1. Log into the root or the PDB using an account that has been granted the DV_OWNER role.

2. Revoke the DV_ADMIN role from the user account whose password needs to change.
3. Connect as a user who has been granted the DV_ACCTMGR role and then change the password for this user.
4. Connect as the DV_OWNER user and then grant the DV_ADMIN role back to the user whose password you changed.

DV_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.6 DV_AUDIT_CLEANUP Audit Trail Cleanup Role

The DV_AUDIT_CLEANUP role is used for purge operations.

Grant the DV_AUDIT_CLEANUP role to any user who is responsible for purging the Database Vault audit trail in a non-unified auditing environment.

Archiving and Purging the Oracle Database Vault Audit Trail explains how to use this role to complete a purge operation.

Privileges Associated with the DV_AUDIT_CLEANUP Role

The DV_AUDIT_CLEANUP role has SELECT and DELETE privileges for three Database Vault-related auditing views.

- SELECT and DELETE on the DVSYS.AUDIT_TRAIL$ table
- SELECT and DELETE on the DVSYS.DV$ENFORCEMENT_AUDIT view
- SELECT and DELETE on the DVSYS.DV$CONFIGURATION_AUDIT view

How Are GRANT and REVOKE Operations Affected by DV_AUDIT_CLEANUP?

By default, this role is granted to the DV_OWNER role with the ADMIN OPTION.

Only a user who has been granted the DV_OWNER role can grant or revoke the DV_AUDIT_CLEANUP role to another user.

DV_AUDIT_CLEANUP Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.7 DV_DATAPUMP_NETWORK_LINK Data Pump Network Link Role

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

Grant the `DV_DATAPUMP_NETWORK_LINK` role to any user who is responsible for conducting the `NETWORK_LINK` transportable Data Pump import operation in an Oracle Database Vault environment.

This role enables the management of the Oracle Data Pump `NETWORK_LINK` transportable import processes to be tightly controlled by Database Vault, but does not change or restrict the way you would normally conduct Oracle Data Pump operations.

Privileges Associated with the DV_DATAPUMP_NETWORK_LINK Role

There are no system privileges associated with the `DV_DATAPUMP_NETWORK_LINK` role, but it does have the `EXECUTE` privilege on `DVSYS` objects.

To find the full list of `DV_DATAPUMP_NETWORK_LINK` object privileges, log into the database instance with sufficient privileges and then enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE =
'DV_DATAPUMP_NETWORK_LINK';
```

Be aware that the `DV_DATAPUMP_NETWORK_LINK` role does not provide a sufficient set of database privileges to conduct `NETWORK_LINK` transportable Data Pump import operation. Rather, the `DV_DATAPUMP_NETWORK_LINK` role is an additional requirement (that is, in addition to the privileges that Oracle Data Pump currently requires) for database administrators to conduct `NETWORK_LINK` transportable Data Pump import operations in an Oracle Database Vault environment.

How Are GRANT and REVOKE Operations Affected by DV_DATAPUMP_NETWORK_LINK?

Only users who have been granted the `DV_OWNER` role can grant or revoke the `DV_DATAPUMP_NETWORK_LINK` role to or from other users.

DV_DATAPUMP_NETWORK_LINK Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the `GRANT ANY ROLE` system privilege can perform `GRANT` and `REVOKE` operations on protected Database Vault roles.

Related Topics

- Using Oracle Data Pump with Oracle Database Vault
  Database administrators can authorize Oracle Data Pump users to work in a Database Vault environment.
- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.8 DV_GOLDENGATE_ADMIN GoldenGate Administrative Role

The `DV_GOLDENGATE_ADMIN` role is used with Oracle GoldenGate.

Grant this role to any user who is responsible for configuring Oracle GoldenGate in an Oracle Database Vault environment.

This enables the management of Oracle GoldenGate processes to be tightly controlled by Database Vault, but does not change or restrict the way an administrator would normally configure Oracle GoldenGate.

Privileges Associated with the DV_GOLDENGATE_ADMIN Role

There are no privileges associated with the `DV_GOLDENGATE_ADMIN` role.

Be aware that the `DV_GOLDENGATE_ADMIN` role does not provide a sufficient set of database privileges for configuring Oracle GoldenGate. Rather, the `DV_GOLDENGATE_ADMIN` role is an additional requirement (that is, in addition to the privileges that Oracle GoldenGate currently requires) for database administrators to configure Oracle GoldenGate in an Oracle Database Vault environment.

How Are GRANT and REVOKE Operations Affected by DV_GOLDENGATE_ADMIN?

Only users who have been granted the `DV_OWNER` role can grant or revoke the `DV_GOLDENGATE_ADMIN` role to or from other users.

DV_GOLDENGATE_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the `GRANT ANY ROLE` system privilege can perform `GRANT` and `REVOKE` operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.
- Privileges for Using Oracle GoldenGate with Oracle Database Vault
  If you want to use Oracle GoldenGate in an Oracle Database Vault environment, then you must have the appropriate privileges.

13.2.9 DV_GOLDENGATE_REDO_ACCESS GoldenGate Redo Log Role

The `DV_GOLDENGATE_REDO_ACCESS` role is used with Oracle GoldenGate.

Grant the DV_GOLDENGATE_REDO_ACCESS role to any user who is responsible for using the Oracle GoldenGate TRANLOGOPTIONS DBLOGREADER method to access redo logs in an Oracle Database Vault environment.

This enables the management of Oracle GoldenGate processes to be tightly controlled by Database Vault, but does not change or restrict the way an administrator would normally configure Oracle GoldenGate.

Privileges Associated with the DV_GOLDENGATE_REDO_ACCESS Role

There are no privileges associated with the DV_GOLDENGATE_REDO_ACCESS role.

Be aware that the DV_GOLDENGATE_REDO_ACCESS role does not provide a sufficient set of database privileges for configuring Oracle GoldenGate. Rather, the DV_GOLDENGATE_REDO_ACCESS role is an additional requirement (that is, in addition to the privileges that Oracle GoldenGate currently requires) for database administrators.

How Are GRANT and REVOKE Operations Affected by DV_GOLDENGATE_REDO_ACCESS?

You cannot grant the DV_GOLDENGATE_REDO_ACCESS role with ADMIN OPTION.

Only users who have been granted the DV_OWNER role can grant or revoke the DV_GOLDENGATE_REDO_ACCESS role to or from other users.

DV_GOLDENGATE_REDO_ACCESS Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.
- Privileges for Using Oracle GoldenGate with Oracle Database Vault
  If you want to use Oracle GoldenGate in an Oracle Database Vault environment, then you must have the appropriate privileges.

13.2.10 DV_MONITOR Database Vault Monitoring Role

The DV_MONITOR role is used for monitoring Oracle Database Vault.

The DV_MONITOR role enables the Oracle Enterprise Manager Cloud Control agent to monitor Oracle Database Vault for attempted violations and configuration issues with realm or command rule definitions.

This role enables Cloud Control to read and propagate realm definitions and command rule definitions between databases.

Privileges Associated with the DV_MONITOR Role

There are no system privileges associated with the DV_MONITOR role, but it does have the SELECT privilege on SYS and DVSYS objects.

In addition, the DV_MONITOR role provides the SELECT privilege on the DBA_DV_POLICY, DBA_DV_POLICY_OWNER, and DBA_DV_POLICY_OBJECT data dictionary views.

To find the full list of DV_MONITOR object privileges, log into the database instance with sufficient (such as DV_OWNER) privileges and then enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_MONITOR';
```

How Are GRANT and REVOKE Operations Affected by DV_MONITOR?

By default, the DV_MONITOR role is granted to the DV_OWNER role and the DBSNMP user.

Only a user who has been granted the DV_OWNER role can grant or revoke the DV_MONITOR role to another user.

DV_MONITOR Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Monitoring Oracle Database Vault
  You can monitor Oracle Database Vault by checking for violations to the Database Vault configurations and by tracking changes to policies.
- Auditing Oracle Database Vault
  You can audit activities in Oracle Database Vault, such as changes to policy configurations.
- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.11 DV_OWNER Database Vault Owner Role

The DV_OWNER role enables you to manage the Oracle Database Vault roles and its configuration.

In this guide, the example account that uses this role is sec_admin_owen.

Privileges Associated with the DV_OWNER Role

The DV_OWNER role has the administrative capabilities that the DV_ADMIN role provides, and the reporting capabilities the DV_SECANALYST role provides.

This role also provides privileges for monitoring Oracle Database Vault. It is created when you install Oracle Database Vault, and has the most privileges on the DVSYS schema. It also has the DV_ADMIN role.

To find the full list of system and object privileges associated with the DV_OWNER role, you can log into the database instance and enter the following queries:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE =
'DV_OWNER';
SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DV_OWNER';
```

When you configure and enable Oracle Database Vault, the DV_OWNER account is created. The user who is granted this role is also granted the ADMIN option and can grant any Oracle Database Vault roles (except DV_ACCTMGR) to any account. Users granted this role also can run Oracle Database Vault reports and monitor Oracle Database Vault.

> **Tip:**
>
> Oracle strongly recommends that you create separate, named account for the DV_OWNER user. This way, if the user is no longer available (for example, they left the company), then you can easily recreate this user account and then grant this user the DV_OWNER role.

How Are GRANT and REVOKE Operations Affected by DV_OWNER?

Anyone with the DV_OWNER role can grant the DV_OWNER and DV_ADMIN roles to another user.

The account granted this role can revoke any granted Database Vault role from another account. Accounts such as SYS or SYSTEM, with the GRANT ANY ROLE system privilege alone (directly granted or indirectly granted using a role) do not have the right to grant or revoke the DV_OWNER role to or from any other database account. Note also that a user with the DV_OWNER role cannot grant or revoke the DV_ACCTMGR role.

Managing Password Changes for Users Who Have the DV_OWNER Role

Before you can change the password for another user who has been granted the DV_OWNER role, you must revoke the DV_OWNER role from that user account.

However, be cautious about revoking the DV_OWNER role. At least one user on your site must have this role granted. If another DV_OWNER user has been granted this role and needs to have their password changed, then you can temporarily revoke DV_OWNER from that user. Note also that if you have been granted the DV_OWNER role, then you can change your own password without having to revoke the role from yourself.

To change the DV_OWNER user password:

1. Log into the root or the PDB using an account that has been granted the DV_OWNER role.
2. Revoke the DV_OWNER role from the user account whose password needs to change.
3. Connect as a user who has been granted the DV_ACCTMGR role and then change the password for this user.
4. Connect as the DV_OWNER user and then grant the DV_OWNER role back to the user whose password you changed.

DV_OWNER Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the `GRANT ANY ROLE` system privilege can perform `GRANT` and `REVOKE` operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.12 DV_PATCH_ADMIN Database Vault Database Patch Role

The `DV_PATCH_ADMIN` role is used for patching operations.

In order to generate all Database Vault-related audit records in accordance with the audit policies specified in the Database Vault metadata as well as Database Vault unified audit policies, run the `DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT` procedure as a user who has been granted the `DV_PATCH_ADMIN` role before using the `DV_PATCH_ADMIN` role. By default, `DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT` is enabled.

Temporarily grant the `DV_PATCH_ADMIN` role to any database administrator who is responsible for performing database patching, typically the user `SYS`. Before this administrator performs the patch operation, run the `DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT` procedure. This procedure enables realm, command rule, and rule set auditing of the actions by users who have been granted the `DV_PATCH_ADMIN` role, in accordance with the existing audit configuration. If you have mixed-mode auditing, then this user's actions are written to the `AUDIT_TRAIL$` table. If you have pure unified auditing enabled, then you should create a unified audit policy to capture this user's actions.

After the patch operation is complete, do not immediately disable the auditing of users who are responsible for performing database patch operations. This way, you can track the actions of the `DV_PATCH_ADMIN` role users. For backwards compatibility, this type of auditing is disabled by default.

Privileges Associated with the DV_PATCH_ADMIN Role

The `DV_PATCH_ADMIN` role does not provide access to any secured data. The common `DV_PATCH_ADMIN` grant is required for database upgrades, and Oracle recommends that this role not be used for any other database administration purpose.

The `DV_PATCH_ADMIN` role a special Database Vault role that does not have any object or system privilege. It is designed to allow the database administrator or the user `SYS` to patch Database Vault enabled databases (for example, applying a database patch without disabling Database Vault). It also enables the database administrator to create users, because some patches may require the need to create new schemas.

Follow these guidelines for managing the `DV_PATCH_ADMIN` role:

- Do not grant the `DV_PATCH_ADMIN` role unless it is required (for example, for a database upgrade).
- Revoke the `DV_PATCH_ADMIN` role grant when the role is no longer needed.

- Review the audit records to monitor activities while the DV_PATCH_ADMIN role was granted.

How Are GRANT and REVOKE Operations Affected by DV_PATCH_ADMIN?

Only a user who has the DV_OWNER role can grant or revoke the DV_PATCH_ADMIN role to and from another user.

DV_PATCH_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Introduction to Auditing
- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.13 DV_POLICY_OWNER Database Vault Owner Role

The DV_POLICY_OWNER role enables database users to manage to a limited degree Oracle Database Vault policies.

Privileges Associated with the DV_POLICY_OWNER Role

The DV_POLICY_OWNER role provides non-Database Vault administrative users the sufficient privileges to enable or disable a Database Vault policy, add or remove authorization to or from a realm, and use the SELECT privilege for the following database views:

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

Only the DV_POLICY_OWNER can query these views. Even users who have the DV_OWNER and DV_ADMIN roles cannot query these views.

The DV_POLICY_OWNER role does not have any system privileges. To find the full list of object privileges that are associated with the DV_POLICY_OWNER role, you can log into the database instance enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE =
'DV_POLICY_OWNER';
```

How Are GRANT and REVOKE Operations Affected by DV_POLICY_OWNER?

Users who have been granted `DV_POLICY_OWNER` role cannot grant or revoke this role to or from other users.

DV_POLICY_OWNER Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the `GRANT ANY ROLE` system privilege can perform `GRANT` and `REVOKE` operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.14 DV_SECANALYST Database Vault Security Analyst Role

The `DV_SECANALYST` role enables users to analyze activities.

Use the `DV_SECANALYST` role to run Oracle Database Vault reports and monitor Oracle Database Vault.

This role is also used for database-related reports. In addition, this role enables you to check the `DVSYS` configuration by querying the `DVSYS` views described in Oracle Database Vault Data Dictionary Views.

Privileges Associated with the DV_SECANALYST Role

There are no system privileges associated with the `DV_SECANALYST` role, but it does have the `SELECT` privilege for some `DVSYS` schema objects and portions of the `SYS` and `SYSMAN` schema objects for reporting on `DVSYS`- and `DVF`-related entities.

In addition, the `DV_SECANALYST` role provides the `SELECT` privilege on the `DBA_DV_POLICY`, `DBA_DV_POLICY_OWNER`, and `DBA_DV_POLICY_OBJECT` data dictionary views.

To find the full list of `DV_SECANALYST` object privileges, log into the database instance with sufficient privileges and then enter the following query:

```
SELECT TABLE_NAME, OWNER, PRIVILEGE FROM DBA_TAB_PRIVS WHERE GRANTEE = 'DV_SECANALYST';
```

How Are GRANT and REVOKE Operations Affected by DV_SECANALYST?

Any account, such as `SYS` or `SYSTEM`, with the `GRANT ANY ROLE` system privilege alone does not have the rights to grant this role to or revoke this role from any other database account.

Only the user with the `DV_OWNER` role can grant or revoke this role to and from another user.

DV_SECANALYST Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database Vault roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

13.2.15 DV_XSTREAM_ADMIN XStream Administrative Role

The DV_XSTREAM_ADMIN role is used for Oracle XStream.

Grant the DV_XSTREAM_ADMIN role to any user who is responsible for configuring Oracle XStream in an Oracle Database Vault environment.

This enables the management of XStream processes to be tightly controlled by Database Vault, but does not change or restrict the way an administrator would normally configure XStream.

Privileges Associated with the DV_XSTREAM_ADMIN Role

There are no privileges associated with the DV_XSTREAM_ADMIN role.

Be aware that the DV_XSTREAM_ADMIN role does not provide a sufficient set of database privileges for configuring XStream. Rather, the DV_XSTREAM_ADMIN role is an additional requirement (that is, in addition to the privileges that XStream currently requires) for database administrators to configure XStream in an Oracle Database Vault environment.

How Are GRANT and REVOKE Operations Affected by DV_XSTREAM_ADMIN?

Only users who have been granted the DV_OWNER role can grant or revoke the DV_XSTREAM_ADMIN role to or from other users.

DV_XSTREAM_ADMIN Status When Oracle Database Vault Security Is Disabled

The protection of all Oracle Database roles is enforced only if Oracle Database Vault is enabled.

If Oracle Database Vault is disabled, then any account with the GRANT ANY ROLE system privilege can perform GRANT and REVOKE operations on protected Database Vault roles.

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.
- Privileges for Using XStream with Oracle Database Vault
  If you want to use XStream in an Oracle Database Vault environment, then you must have the appropriate privileges.

13.3 Oracle Database Vault Accounts Created During Registration

The accounts that you create during registration enable Oracle Database Vault to adhere to separation of duty concepts and provide flexibility for users based on the tasks they perform.

- About Oracle Database Vault Accounts Created During Registration
  You must create accounts for the Oracle Database Vault Owner and Oracle Database Vault Account Manager during the registration process.
- Database Accounts Used by Oracle Database Vault
  Oracle Database Vault provides accounts that provide access to system and object privileges, and Oracle Label Security.
- Model Oracle Database Vault Database Accounts
  You can create different database accounts to implement the separation of duties requirements for Oracle Database Vault.

13.3.1 About Oracle Database Vault Accounts Created During Registration

You must create accounts for the Oracle Database Vault Owner and Oracle Database Vault Account Manager during the registration process.

You must supply an account name and password for the Oracle Database Vault Owner accounts during installation. Creating an Oracle Database Vault Account Manager is optional but strongly recommended for better separation of duty.

The Oracle Database Vault Owner account is granted the `DV_OWNER` role. This account can manage Oracle Database Vault roles and configuration.

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

If you choose not to create the Oracle Database Vault Account Manager account during installation, then both the `DV_OWNER` and `DV_ACCTMGR` roles are granted to the Oracle Database Vault Owner user account.

13.3.2 Database Accounts Used by Oracle Database Vault

Oracle Database Vault provides accounts that provide access to system and object privileges, and Oracle Label Security.

The following table lists the Oracle Database Vault database accounts that are needed in addition to the accounts that you create during installation.

Table 13-1 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

Table 13-1 (Cont.) Database Accounts Used by Oracle Database Vault

Database Account	Roles and Privileges	Description
DVF	A limited set of system privileges are provided to support Oracle Database Vault. The ability to create a session with this account is revoked at the end of the installation, and the account is locked.	Owner of the Oracle Database Vault functions that are created to retrieve factor identities
LBACSYS	This account is created when you install Oracle Label Security by using the Oracle Universal Installer custom installation option. (It is not created when you install Oracle Database Vault.) Do not drop or re-create this account. If you plan to integrate a factor with an Oracle Label Security policy, you must assign this user as the owner of the realm that uses this factor.	Owner of the Oracle Label Security schema

Related Topics

- Using Oracle Database Vault Factors with Oracle Label Security Policies
  To enhance security, you can integrate Oracle Database Vault factors with Oracle Label Security policies.

13.3.3 Model Oracle Database Vault Database Accounts

You can create different database accounts to implement the separation of duties requirements for Oracle Database Vault.

The following table lists some model database accounts that can act as a guide. (The accounts listed in this table serve as a guide to implementing Oracle Database Vault roles. These are not actual accounts that are created during installation.)

Table 13-2 Model Oracle Database Vault Database Accounts

Database Account	Roles and Privileges	Description
EBROWN	DV_OWNER (with DV_ADMIN and DV_SECANALYST)	Account that is the realm owner for the Oracle Database Vault realm. This account can: • Run DVSYS packages. • Grant privileges on the DVSYS schema objects. • Select objects in the DVSYS schema. • Monitor Oracle Database Vault activity. • Run reports on the Oracle Database Vault configuration.

Table 13-2 (Cont.) Model Oracle Database Vault Database Accounts

Database Account	Roles and Privileges	Description
JGODFREY	DV_ACCTMGR	Account for administration of database accounts and profiles. This account can: • Create, alter, and drop users. • Create, alter, and drop profiles. • Grant and revoke the CREATE SESSION privilege. • Grant and revoke the DV_ACCTMGR role, but only if this account was created during the Database Vault installation (this account is created with the ADMIN option). • Grant and revoke the CONNECT role. **Note:** This account cannot create roles, or grant the RESOURCE or DBA roles.
RLAYTON	DV_ADMIN (with DV_SECANALYST)	Account to serve as the access control administrator. This account can: • Run DVSYS packages. • Monitor Oracle Database Vault activity. • Run reports on the Oracle Database Vault configuration. **Note:** This account cannot directly update the DVSYS tables.
PSMYTHE	DV_SECANALYST	Account for running Oracle Database Vault reports

Related Topics

- Configuring Oracle Database Vault Accounts as Enterprise User Accounts
  You can configure existing Oracle Database Vault user accounts as enterprise user accounts in a PDB.
- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the DV_OWNER and DV_ACCTMGR roles.

13.4 Backup Oracle Database Vault Accounts

As a best practice, you should maintain backup accounts for the DV_OWNER and DV_ACCTMGR roles.

The Oracle Database Vault registration process entails creating both day-to-day and backup accounts for the DV_OWNER and DV_ACCTMGR roles. You should keep and maintain these accounts as a safety measure in case a user who has been granted one of these roles forgets their password or leaves the organization. Then you can log in to the backup account to recover the password or grant the role to a new account. These should be only used as a backup account kept safe in a privileged account management system or an organization break-glass (or emergency password recovery) system. When you grant a user one of these roles, include the WITH ADMIN OPTION clause in the GRANT statement.

Because of the strong separation of duty that Oracle Database Vault implements, loss of access to the DV_OWNER account will force you to rebuild the database. The SYS account cannot override the DV_OWNER account

Related Topics

- Resetting Oracle Database Vault Account Passwords
  Backup accounts can help you reset lost passwords for users who have been granted the DV_OWNER and DV_ACCTMGR roles.

14
Oracle Database Vault Realm APIs

The DBMS_MACADM PL/SQL package enables you to configure Oracle Database Vault realms.

Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures. For constants that you can use with these procedures, see Table 20-1 for more information.

- ADD_AUTH_TO_REALM Procedure
  The ADD_AUTH_TO_REALM procedure authorizes a user or role to access a realm as an owner or a participant. You can authenticate both common and local realms.
- ADD_OBJECT_TO_REALM Procedure
  The ADD_OBJECT_TO_REALM procedure registers a set of objects for realm protection.
- CREATE_REALM Procedure
  The CREATE_REALM procedure creates both common and local realms.
- DELETE_AUTH_FROM_REALM Procedure
  The DELETE_AUTH_FROM_REALM procedure removes the authorization of a user or role to access a realm.
- DELETE_OBJECT_FROM_REALM Procedure
  The DELETE_OBJECT_FROM_REALM procedure removes a set of objects from realm protection.
- DELETE_REALM Procedure
  The DELETE_REALM procedure deletes a realm, including its related configuration information that specifies who is authorized and what objects are protected.
- DELETE_REALM_CASCADE Procedure
  The DELETE_REALM_CASCADE procedure deletes a realm, including its related Database Vault configuration information that specifies who is authorized and the objects that are protected.
- RENAME_REALM Procedure
  The RENAME_REALM procedure renames a realm; the name change takes effect everywhere the realm is used.
- UPDATE_REALM Procedure
  The UPDATE_REALM procedure updates a realm.
- UPDATE_REALM_AUTH Procedure
  The UPDATE_REALM_AUTH procedure updates the authorization of a user or role to access a realm.

14.1 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. You can authenticate both common and local realms.

Optionally, you can specify a rule set that must be checked before allowing the authorization to be enabled.

Syntax

```
DBMS_MACADM.ADD_AUTH_TO_REALM(
 realm_name    IN VARCHAR2,
 grantee       IN VARCHAR2,
 rule_set_name IN VARCHAR2,
 auth_options  IN NUMBER
 auth_scope    IN NUMBER DEFAULT);
```

Parameters

Table 14-1 ADD_AUTH_TO_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view.
grantee	User or role name to authorize as an owner or a participant. To find the existing users and roles in the current database instance, query the DBA_USERS and DBA_ROLES views. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view. To find existing secure application roles used in privilege management, query the DBA_DV_ROLE view.
rule_set_name	Optional. The rule set to check during runtime. The realm authorization is enabled only if the rule set evaluates to TRUE. To find the available rule sets, query the DBA_DV_RULE_SET view.
auth_options	Optional. Specify one of the following options to authorize the realm: • DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT: Participant. This account or role allows system or direct privileges to access, manipulate, and create objects protected by the realm, provided these rights have been granted using the standard Oracle Database privilege grant process. (Default) • DBMS_MACUTL.G_REALM_AUTH_OWNER: Owner. This account or role has the same authorization as the realm participant, plus the authorization to grant or revoke realm-secured roles and privileges on realm-protected objects. See Related Topics for more information about participants and owners.
auth_scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) to authorize the realm locally in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) to authorize the realm in the application root

Examples

The following example authorizes user SYSADM as a participant in the Performance Statistics Realm. Because the default is to authorize the user as a participant, the auth_options parameter can be omitted.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
```

```
  realm_name  => 'Performance Statistics Realm',
  grantee     => 'SYSADM');
END;
/
```

This example sets user `SYSADM` as the owner of the Performance Statistics Realm.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name   => 'Performance Statistics Realm',
  grantee      => 'SYSADM',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

The next example triggers the Check Conf Access rule set before allowing user `SYSADM` to act as the owner of the Performance Statistics Realm.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    => 'Performance Statistics Realm',
  grantee       => 'SYSADM',
  rule_set_name => 'Check Conf Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

This example shows how to commonly grant the common user `C##HR_ADMIN` access to the common realm HR Statistics Realm. The user running this procedure must be in the CDB root, and the rule set must be a common rule set residing in the application root.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    => 'HR Statistics Realm',
  grantee       => 'C##HR_ADMIN',
  rule_set_name => 'Check Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER,
 auth_scope    => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This example shows how to locally grant the common user `C##HR_CLERK` access to the common realm HR Statistics Realm. The user running this procedure must be in the same PDB in which the authorization applies. To find the existing PDBs query the `DBA_PDBS` data dictionary view. The rule set must be a local rule set.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name    => 'HR Statistics Realm',
  grantee       => 'C##HR_CLERK',
  rule_set_name => 'Check Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER,
  auth_scope    => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

Related Topics

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

14.2 ADD_OBJECT_TO_REALM Procedure

The ADD_OBJECT_TO_REALM procedure registers a set of objects for realm protection.

Syntax

```
DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   IN VARCHAR2,
  object_owner IN VARCHAR2,
  object_name  IN VARCHAR2,
  object_type  IN VARCHAR2);
```

Parameters

Table 14-2 ADD_OBJECT_TO_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view.
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. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view.
object_name	Object name. (The wildcard % is allowed.) You can also use the DBMS_MACUTL.G_ALL_OBJECT constant. To find the available objects, query the ALL_OBJECTS view. To find objects that are secured by existing realms, query the DBA_DV_REALM_OBJECT view.
object_type	Object type, such as TABLE, INDEX, or ROLE. (The wildcard % is allowed.) You can also use the DBMS_MACUTL.G_ALL_OBJECT constant.

Example

```
BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name    => 'HR Apps',
  object_owner  => '%',
  object_name   => 'HR_SELECT_ROLE',
  object_type   => 'ROLE');
END;
/
```

Related Topics

- About Realm-Secured Objects
  Realm-secured objects define the territory—a set of schema and database objects and roles—that a realm protects.

14.3 CREATE_REALM Procedure

The `CREATE_REALM` procedure creates both common and local realms.

After you create the realm, use the following procedures to complete the realm definition:

- `ADD_OBJECT_TO_REALM` procedure registers one or more objects for the realm.
- `ADD_AUTH_TO_REALM` procedure authorizes users or roles for the realm.

Syntax

```
DBMS_MACADM.CREATE_REALM(
 realm_name    IN VARCHAR2,
 description   IN VARCHAR2,
 enabled       IN VARCHAR2,
 audit_options IN NUMBER,
 realm_type    IN NUMBER DEFAULT,
 realm_scope   IN NUMBER DEFAULT
 pl_sql_stack  IN BOOLEAN DEFAULT);
```

Parameters

Table 14-3 CREATE_REALM Parameters

Parameter	Description
`realm_name`	Realm name, up to 128 characters in mixed-case. Oracle suggests that you use the name of the protected application as the realm name (for example, `hr_app` for an human resources application). This parameter is mandatory. To find the existing realms in the current database instance, query the `DBA_DV_REALM` view.
`description`	Description of the purpose of the realm, up to 1024 characters in mixed-case. This parameter is optional. You may want to include a description for the business objective of the given application protection and document all other security policies that compliment the realm's protection. Also document who is authorized to the realm, for what purpose, and any possible emergency authorizations.
`enabled`	Specify one of the following mandatory options to set the status of the realm: • `DBMS_MACUTL.G_YES` or `'y'` to enable realm checking (default) • `DBMS_MACUTL.G_NO` or `'n'` to disable all realm checking, including the capture of violations in the simulation log • `DBMS_MACUTL.G_SIMULATION` or `'s'` to enable SQL statements to execute but capture violations in the simulation log
`audit_options`	Omit, or specify `DBMS_MACUTL.G_REALM_AUDIT_OFF` (default). Starting with Oracle Database release 23c, traditional auditing is desupported. Only the `DBMS_MACUTL.G_REALM_AUDIT_OFF` setting is available for the `audit_options` parameter. To audit a realm, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a realm.

Table 14-3 (Cont.) CREATE_REALM Parameters

Parameter	Description
realm_type	Specify one of the following options: • 0: Disables mandatory realm checking. • 1: Enables mandatory realm checking for realm objects. Only realm owners or realm participants will have access to objects in a realm. Object owners and object-privileged users who are not realm owners or participants will have no access. See also Related Topics.
realm_scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the realm must be local in the current PDB. • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the realm must be in the application root. This setting duplicates the realm in all of the associated PDBs. If you create the common realm in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record. The default is FALSE.

Examples

The following example shows how to create a realm that is enabled and uses mandatory realm checking, and records the PL/SQL stack.

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name     => 'HR Apps',
  description    => 'Realm to protect the HR schema',
  enabled        => DBMS_MACUTL.G_YES,
  audit_options  => DBMS_MACUTL.G_REALM_AUDIT_OFF,
  realm_type     => 1,
  pl_sql_stack   => TRUE);
END;
/
```

This example shows how to create a variation of the preceding example, but as a common realm located in the application root. The user who creates this realm must be a common user and must run the procedure in the application root.

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name     => 'HR Apps',
  description    => 'Realm to protect the HR schema',
  enabled        => DBMS_MACUTL.G_YES,
  audit_options  => DBMS_MACUTL.G_REALM_AUDIT_OFF,
  realm_type     => 1,
  realm_scope    => DBMS_MACUTL.G_SCOPE_COMMON);
```

```
END;
/
```

This example shows how to create a local version of the preceding example. The user who creates this realm must be in the PDB in which the realm will reside. To find existing PDBs, query the DBA_PDBS data dictionary view.

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name     => 'HR Apps',
  description    => 'Realm to protect the HR schema',
  enabled        => DBMS_MACUTL.G_YES,
  audit_options  => DBMS_MACUTL.G_REALM_AUDIT_OFF,
  realm_type     => 1,
  realm_scope    => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

Related Topics

- Mandatory Realms to Restrict User Access to Objects within a Realm
  By default, users who own or have object privileges are allowed to access realm-protected objects without explicit realm authorization.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

14.4 DELETE_AUTH_FROM_REALM Procedure

The DELETE_AUTH_FROM_REALM procedure removes the authorization of a user or role to access a realm.

Syntax

```
DBMS_MACADM.DELETE_AUTH_FROM_REALM(
 realm_name    IN VARCHAR2,
 grantee       IN VARCHAR2,
 auth_scope    IN NUMBER DEFAULT);
```

Parameters

Table 14-4 DELETE_AUTH_FROM_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view.
grantee	User or role name. To find the authorization of a particular user or role, query the DVA_DV_REALM_AUTH view.

Table 14-4 (Cont.) DELETE_AUTH_FROM_REALM Parameters

Parameter	Description
auth_scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the realm was authorized locally in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2 if the realm was authorized in the application root

Example

```
BEGIN
DBMS_MACADM.DELETE_AUTH_FROM_REALM(
 realm_name     => 'HR Apps',
 grantee        => 'PSMITH',
 auth_scope     => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

14.5 DELETE_OBJECT_FROM_REALM Procedure

The DELETE_OBJECT_FROM_REALM procedure removes a set of objects from realm protection.

Syntax

```
DBMS_MACADM.DELETE_OBJECT_FROM_REALM(
  realm_name   IN VARCHAR2,
  object_owner IN VARCHAR2,
  object_name  IN VARCHAR2,
  object_type  IN VARCHAR2);
```

Parameters

Table 14-5 DELETE_OBJECT_FROM_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view.
object_owner	The owner of the object that was added to the realm. To find the available users, query the DBA_USERS view.
object_name	Object name. (The wildcard % is allowed.) You can also use the DBMS_MACUTL.G_ALL_OBJECT constant. To find objects that are secured by existing realms, query the DBA_DV_REALM_OBJECT view. See also Related Topics.

Table 14-5 (Cont.) DELETE_OBJECT_FROM_REALM Parameters

Parameter	Description
object_type	Object type, such as TABLE, INDEX, or ROLE. (The wildcard % is allowed.) You can also use the DBMS_MACUTL.G_ALL_OBJECT constant. See also Related Topics.

Example

```
BEGIN
 DBMS_MACADM.DELETE_OBJECT_FROM_REALM(
  realm_name     => 'HR Apps',
  object_owner   => '%',
  object_name    => 'HR_SELECT_ROLE',
  object_type    => '%');
END;
/
```

Related Topics

- About Realm-Secured Objects
  Realm-secured objects define the territory—a set of schema and database objects and roles—that a realm protects.

14.6 DELETE_REALM Procedure

The DELETE_REALM procedure deletes a realm, including its related configuration information that specifies who is authorized and what objects are protected.

This procedure does not delete the actual database objects or users.

To find users who are authorized for the realm, query the DBA_DV_REALM_AUTH view. To find the objects that are protected by the realm, query the DBA_DV_REALM_OBJECT view.

Syntax

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

Parameters

Table 14-6 DELETE_REALM Parameter

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view.

Example

```
EXEC DBMS_MACADM.DELETE_REALM ('HR Apps');
```

14.7 DELETE_REALM_CASCADE Procedure

The `DELETE_REALM_CASCADE` procedure deletes a realm, including its related Database Vault configuration information that specifies who is authorized and the objects that are protected.

The `DBA_DV_REALM_AUTH` view lists who is authorized in the realm and the `DBA_DV_REALM_OBJECT` view lists the protected objects.

It does not delete the actual database objects or users. This procedure works the same as the `DELETE_REALM` procedure. (In previous releases, these procedures were different, but now they are the same. Both are retained for earlier compatibility.) To find a listing of the realm-related objects, query the `DBA_DV_REALM` view. To find its authorizations, query `DBA_DV_REALM_AUTH`.

Syntax

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

Parameters

Table 14-7 DELETE_REALM_CASCADE Parameter

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DBA_DV_REALM` view.

Example

```
BEGIN
 DBMS_MACADM.RENAME_REALM(
  realm_name  => 'HR Apps',
  new_name    => 'HR and HCM Apps');
END;
/
```

14.8 RENAME_REALM Procedure

The `RENAME_REALM` procedure renames a realm; the name change takes effect everywhere the realm is used.

Syntax

```
DBMS_MACADM.RENAME_REALM(
 realm_name  IN VARCHAR2,
 new_name    IN VARCHAR2);
```

Parameters

Table 14-8 RENAME_REALM Parameters

Parameter	Description
realm_name	Current realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view.
new_name	New realm name, up to 128 characters in mixed-case.

Example

```
BEGIN
 DBMS_MACADM.RENAME_REALM(
  realm_name => 'Performance Statistics Realm',
  new_name   => 'Sector 2 Performance Statistics Realm');
END;
/
```

14.9 UPDATE_REALM Procedure

The UPDATE_REALM procedure updates a realm.

To find information about the current settings for a realm, query the DVSYS.DV$REALM view.

When you update the audit options in a realm, the existing traditional audit records are disabled. You must create a unified audit policy to capture new audit records.

Syntax

```
DBMS_MACADM.UPDATE_REALM(
 realm_name    IN VARCHAR2,
 description   IN VARCHAR2,
 enabled       IN VARCHAR2,
 audit_options IN NUMBER DEFAULT NULL,
 realm_type    IN NUMBER DEFAULT NULL
 pl_sql_stack  IN BOOLEAN DEFAULT NULL);
```

Parameters

Table 14-9 UPDATE_REALM Parameters

Parameter	Description
realm_name	Realm name. To find the existing realms in the current database instance, query the DBA_DV_REALM view.
description	Description of the purpose of the realm, up to 1024 characters in mixed-case.

Table 14-9 (Cont.) UPDATE_REALM Parameters

Parameter	Description
enabled	Specify one of the following options to set the status of the realm: • DBMS_MACUTL.G_YES or 'y' to enable realm checking • DBMS_MACUTL.G_NO or 'n' to disable all realm checking, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log The default for enabled is the previously set value, which you can find by querying the DBA_DV_REALM data dictionary view.
audit_options	Perform one of the following actions: • Set DBMS_MACUTL.G_REALM_AUDIT_OFF to disable traditional auditing for this realm policy. • Leave the current existing audit_options setting as is if you want to continue using this traditional auditing setting. To do so, you can specify NULL, or specify the same setting as the current existing audit_options for this realm, or simply do not specify audit_options (the default is NULL). Starting with Oracle Database 23c, traditional auditing is desupported. To audit a realm, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a realm.
realm_type	If you do not specify the realm_type parameter, then Oracle Database Vault does not update the current realm_type setting. Specify one of the following options: • 0: Sets the realm to be a regular realm, which does not have mandatory realm checking. • 1: Enables mandatory realm checking for realm objects. Only realm owners or realm participants will have access to objects in a realm. Object owners and object-privileged users who are not realm owners or participants will have no access. See also Related Topics.
pl_sql_stack	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_REALM(
  realm_name      => 'HR Apps',
  description     => 'Realm to protect the HR schema',
  enabled         => DBMS_MACUTL.G_YES,
  audit_options   => DBMS_MACUTL.G_REALM_AUDIT_OFF,
  realm_type      => 1);
END;
/
```

If you did not make any modifications to the audit_options parameter of your realm, then the existing traditional audit policy will still be in place. If you updated the audit_options parameter of the realm, then this auditing will be disabled because

traditional auditing is desupported starting in release 23c. To capture new audit records, create and enable a unified audit policy. For example:

```
CREATE AUDIT POLICY hr_app_aud_pol
 ACTIONS COMPONENT=DV Realm Violation ON "HR Apps";

AUDIT POLICY hr_app_aud_pol EXCEPT psmith;
```

You can view the audit records by querying the `UNIFIED_AUDIT_TRAIL` data dictionary view. See How Oracle *Oracle Database Security Guide* for how this works.

Related Topics

- Mandatory Realms to Restrict User Access to Objects within a Realm
  By default, users who own or have object privileges are allowed to access realm-protected objects without explicit realm authorization.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

14.10 UPDATE_REALM_AUTH Procedure

The `UPDATE_REALM_AUTH` procedure updates the authorization of a user or role to access a realm.

Syntax

```
DBMS_MACADM.UPDATE_REALM_AUTH(
 realm_name    IN VARCHAR2,
 grantee       IN VARCHAR2,
 rule_set_name IN VARCHAR2,
 auth_options  IN NUMBER,
 auth_scope    IN NUMBER DEFAULT);
```

Parameters

Table 14-10 UPDATE_REALM_AUTH Parameters

Parameter	Description
`realm_name`	Realm name. To find the existing realms in the current database instance, query the `DBA_DV_REALM` view.
`grantee`	User or role name. To find the available users and roles in the current database instance, query the DBA_USERS and DBA_ROLES data dictionary views. To find the authorization of a particular user or role, query the `DVA_DV_REALM_AUTH` view. To find existing secure application roles used in privilege management, query the `DBA_DV_ROLE` view.

Table 14-10 (Cont.) UPDATE_REALM_AUTH Parameters

Parameter	Description
rule_set_name	Optional. A rule set to check during runtime. The realm authorization is enabled only if the rule set evaluates to TRUE. To find the available rule sets, query the DBA_DV_RULE_SET view. To find rules that are associated with the rule sets, query the DBA_DB_RULE_SET_RULE view.
auth_options	Optional. Specify one of the following options to authorize the realm: • DBMS_MACUTL.G_REALM_AUTH_PARTICIPANT: Participant. This account or role provides system or direct privileges to access, manipulate, and create objects protected by the realm, provided these rights have been granted using the standard Oracle Database privilege grant process. • DBMS_MACUTL.G_REALM_AUTH_OWNER: Owner. This account or role has the same authorization as the realm participant, plus the authorization to grant or revoke realm-secured roles and privileges on realm-protected objects. A realm can have multiple owners. The default for auth_options value is the previously set value, which you can find by querying the DBA_DV_REALM_AUTH data dictionary view.
realm_auth	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the realm is authorized locally in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the realm is authorized in the application root

Example

```
BEGIN
 DBMS_MACADM.UPDATE_REALM_AUTH(
  realm_name    => 'HR Apps',
  grantee       => 'HR_SELECT_ROLE',
  rule_set_name => 'Check Conf Access',
  auth_options  => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

15
Oracle Database Vault Rule Set APIs

You can use the `DBMS_MACADM` PL/SQL package and a set of Oracle Database Vault rule functions to manage rule sets.

- DBMS_MACADM Rule Set Procedures
  The `DBMS_MACADM` rule set procedures enable you to configure both rule sets and individual rules that go within these rule sets.
- Oracle Database Vault PL/SQL Rule Set Functions
  Oracle Database Vault provides functions to use in rule sets to inspect the SQL statement that the rule set protects.

15.1 DBMS_MACADM Rule Set Procedures

The `DBMS_MACADM` rule set procedures enable you to configure both rule sets and individual rules that go within these rule sets.

Only users who have been granted the `DV_OWNER` or `DV_ADMIN` role can use these procedures.

- ADD_RULE_TO_RULE_SET Procedure
  The `ADD_RULE_TO_RULE_SET` procedure adds rule to a rule set; you can enable having the rule checked when the rule set is evaluated.
- CREATE_RULE Procedure
  The `CREATE_RULE` procedure creates both common and local rules, which afterward, can be added to a rule set.
- CREATE_RULE_SET Procedure
  The `CREATE_RULE_SET` procedure creates a rule set.
- DELETE_RULE Procedure
  The `DELETE_RULE` procedure deletes a rule.
- DELETE_RULE_FROM_RULE_SET Procedure
  The `DELETE_RULE_FROM_RULE_SET` procedure deletes a rule from a rule set.
- DELETE_RULE_SET Procedure
  The `DELETE_RULE_SET` procedure deletes a rule set.
- RENAME_RULE Procedure
  The `RENAME_RULE` procedure renames a rule and causes the name change to take effect everywhere the rule is used
- RENAME_RULE_SET Procedure
  The `RENAME_RULE_SET` procedure renames a rule set and causes the name change to take effect everywhere the rule set is used.
- UPDATE_RULE Procedure
  The `UPDATE_RULE` procedure updates a rule.
- UPDATE_RULE_SET Procedure
  The `UPDATE_RULE_SET` procedure updates a rule set.

Related Topics

- Configuring Rule Sets
  Rule sets group one or more rules together; the rules determine whether a user can perform an action on an object.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

15.1.1 ADD_RULE_TO_RULE_SET Procedure

The `ADD_RULE_TO_RULE_SET` procedure adds rule to a rule set; you can enable having the rule checked when the rule set is evaluated.

Syntax

```
DBMS_MACADM.ADD_RULE_TO_RULE_SET(
 rule_set_name  IN VARCHAR2,
 rule_name      IN VARCHAR2,
 rule_order     IN NUMBER,
 enabled        IN VARCHAR2,
 scope          IN NUMBER DEFAULT);
```

Parameters

Table 15-1 ADD_RULE_TO_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view.
`rule_name`	Rule to add to the rule set. To find existing rules, query the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, query `DBA_DV_RULE_SET_RULE`.
`rule_order`	Does not apply to this release, but you must include a value for the `ADD_RULE_TO_RULE_SET` procedure to work. Enter `1`.
`enabled`	Optional. Determines whether the rule should be checked when the rule set is evaluated. Possible values are: • `DBMS_MACUTL.G_YES` (default). Enables the rule to be checked during the rule set evaluation. • `DBMS_MACUTL.G_NO` Prevents the rule from being checked during the rule set evaluation. See also Related Topics.
`scope`	Determines how to execute this procedure. The default is local. Options are as follows: • `DBMS_MACUTL.G_SCOPE_LOCAL` (or `1`) if the rule and rule set are local in the current PDB • `DBMS_MACUTL.G_SCOPE_COMMON` (or `2`) if the rule and rule set are in the application root

Examples

The following example adds a rule to a rule set, and by omitting the `enabled` parameter, automatically enables the rule to be checked when the rule set is evaluated.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Restrict DROP TABLE operations',
  rule_order    => 1);
END;
/
```

This example adds the rule to the rule set but disables rule checking.

```
BEGIN
 DBMS_MACADM.ADD_RULE_TO_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Check UPDATE operations',
  rule_order    => 1,
  enabled       => DBMS_MACUTL.G_NO);
END;
/
```

Related Topics

- DBMS_MACUTL Constants
  You can use a set of constants, available in the `DBMS_MACUTL` PL/SQL package.

15.1.2 CREATE_RULE Procedure

The `CREATE_RULE` procedure creates both common and local rules, which afterward, can be added to a rule set.

Syntax

```
DBMS_MACADM.CREATE_RULE(
 rule_name  IN VARCHAR2,
 rule_expr  IN VARCHAR2
 scope      IN NUMBER DEFAULT,
 is_static  IN BOOLEAN DEFAULT FALSE);
```

Parameters

Table 15-2 CREATE_RULE Parameters

Parameter	Description
`rule_name`	Rule name, up to 128 characters in mixed-case. Spaces are allowed. To find existing rules in the current database instance, query the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, query `DBA_DV_RULE_SET_RULE`.

Table 15-2 (Cont.) CREATE_RULE Parameters

Parameter	Description
rule_expr	PL/SQL BOOLEAN expression. If the expression contains quotation marks, do not use double quotation marks. Instead, use two single quotation marks. Enclose the entire expression within single quotation marks. For example: 'TO_CHAR(SYSDATE,''HH24'') = ''12''''
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule is in the application root
is_static	Specifies whether the evaluation of a rule should be static or dynamic. When a rule is specified as static, then the result of the first rule evaluation in a session will be cached and used whenever the rule needs to be evaluated again in the session.

Examples

The following example shows how to create a local rule expression that checks if the current session user is SYSADM. The user running this procedure must be in the same PDB in which the rule and its rule set reside. To find the existing PDBs, run the show pdbs command. The rule and rule set must be local.

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

This example shows a multitenant environment common version of the preceding example. The user running this procedure must be in the CDB root, and the rule and its associated rule set must be common. The rule will reside in the application root.

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

This example shows how to create a rule expression that uses the public standalone function OLS_LABEL_DOMINATES to find if the session label of the hr_ols_pol Oracle Label Security policy dominates or is equal to the hs label. The value 0 indicates if it is false. (To check if it is equal, you would specify 1.) The scope (default is local) and is_static (default is FALSE) parameters are omitted; their default values will be used.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name  => 'Check OLS Factor',
  rule_expr  => 'OLS_LABEL_DOMINATES(''hr_ols_pol'', ''hs'') = 1');
END;
/
```

Related Topics

- Creating a Rule
  You can create a new rule or use the default Oracle Database Vault rules.

15.1.3 CREATE_RULE_SET Procedure

The `CREATE_RULE_SET` procedure creates a rule set.

After you create a rule set, you can use the `CREATE_RULE` and `ADD_RULE_TO_RULE_SET` procedures to create and add rules to the rule set.

Syntax

```
DBMS_MACADM.CREATE_RULE_SET(
 rule_set_name    IN VARCHAR2,
 description      IN VARCHAR2,
 enabled          IN VARCHAR2,
 eval_options     IN NUMBER,
 audit_options    IN NUMBER,
 fail_options     IN NUMBER,
 fail_message     IN VARCHAR2,
 fail_code        IN NUMBER,
 handler_options  IN NUMBER,
 handler          IN VARCHAR2,
 is_static        IN BOOLEAN DEFAULT,
 scope            IN NUMBER DEFAULT);
```

Parameters

Table 15-3 CREATE_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name, up to 128 characters in mixed-case. Spaces are allowed. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view.
`description`	Description of the purpose of the rule set, up to 1024 characters in mixed-case.
`enabled`	`DBMS_MACUTL.G_YES` (Yes) enables the rule set; `DBMS_MACUTL.G_NO` (No) disables it. The default is `DBMS_MACUTL.G_YES`.
`eval_options`	If you plan to assign multiple rules to the rule set, enter one of the following settings: • `DBMS_MACUTL.G_RULESET_EVAL_ALL`: All rules in the rule set must evaluate to true for the rule set itself to evaluate to true (default). • `DBMS_MACUTL.G_RULESET_EVAL_ANY`: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to true.

Table 15-3 (Cont.) CREATE_RULE_SET Parameters

Parameter	Description
audit_options	Omit, or specify DBMS_MACUTL.G_RULESET_AUDIT_OFF (default). Starting with Oracle Database 23c, traditional auditing is desupported. To audit a rule set, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a rule set.
fail_options	Options for reporting errors: • DBMS_MACUTL.G_RULESET_FAIL_SHOW: Shows an error message (default) • DBMS_MACUTL.G_RULESET_FAIL_SILENT: Does not show an error message
fail_message	Enter an error message for failure, up to 80 characters in mixed-case, to associate with the fail code you specify for fail_code.
fail_code	Enter a number in the range of -20000 to -20999 or 20000 to 20999 to associate with the fail_message parameter.
handler_options	Select one of the following settings: • DBMS_MACUTL.G_RULESET_HANDLER_OFF: Disables error handling (default) • DBMS_MACUTL.G_RULESET_HANDLER_FAIL: Calls handler on rule set failure • DBMS_MACUTL.G_RULESET_HANDLER_SUCCESS: Calls handler on rule set success
handler	Name of the PL/SQL function or procedure that defines the custom event handler logic.
is_static	Optional. Determines how often a rule set is evaluated when it is accessed. The default is FALSE. • TRUE: The rule set is evaluated once during the user session. After that, the value is re-used. • FALSE: The rule set is evaluated every time.
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule set is to be local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule set is to be in the application root

Examples

The following example creates a rule set that is enabled and is set so that at least one rule must evaluate to true for the rule set itself to evaluate to true. It does not show error messages but uses the fail code 20461 to track failures. It also uses a handler to send email alerts to the appropriate users if their are violations to the rule set.

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
```

```
  fail_message     => 'Configuration failed; check settings',
  fail_code        => 20461,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler          => 'dbavowner.email_alert',
  is_static        => TRUE);
END;
/
```

This rule set uses no fail messages or fail codes, nor does it use any handlers. This rule set will be in the application root of a multitenant environment, so the user running this procedure must be in the application root. Any rules or command rules that are associated with this rule set must be common.

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
 rule_set_name    => 'Check_HR_Access',
 description      => 'Checks for failed access attempts to the HR schema',
 enabled          => DBMS_MACUTL.G_YES,
 eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
 audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
 fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
 fail_message     => '',
 fail_code        => '',
 handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
 handler          => '',
 is_static        => TRUE,
 scope            => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This rule set is a local version of the preceding rule set. The user who creates this rule set must be in the PDB in which this rule set will reside. To find the existing PDBs, query the `DBA_PDBS` data dictionary view. Any rules or command rules that are associated with this rule set must be local.

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
 rule_set_name    => 'Check_HR_Access',
 description      => 'Checks for failed access attempts to the HR schema',
 enabled          => DBMS_MACUTL.G_YES,
 eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
 audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
 fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
 fail_message     => '',
 fail_code        => '',
 handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
 handler          => '',
 is_static        => TRUE,
 scope            => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

Related Topics

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

15.1.4 DELETE_RULE Procedure

The DELETE_RULE procedure deletes a rule.

Syntax

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

Parameter

Table 15-4 DELETE_RULE Parameter

Parameter	Description
rule_name	Rule name. To find existing rules in the current database instance, query the DBA_DV_RULE view. To find rules that have been associated with rule sets, query DBA_DV_RULE_SET_RULE.

Example

```
EXEC DBMS_MACADM.DELETE_RULE('Check UPDATE operations');
```

15.1.5 DELETE_RULE_FROM_RULE_SET Procedure

The DELETE_RULE_FROM_RULE_SET procedure deletes a rule from a rule set.

Syntax

```
DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
 rule_set_name IN VARCHAR2,
 rule_name     IN VARCHAR2);
```

Parameters

Table 15-5 DELETE_RULE_FROM_RULE_SET Parameters

Parameter	Description
rule_set_name	Rule set name. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view.
rule_name	Rule to remove from the rule set. To find existing rules in the current database instance, query the DBA_DV_RULE view. To find rules that have been associated with rule sets, query DBA_DV_RULE_SET_RULE.

Example

```
BEGIN
 DBMS_MACADM.DELETE_RULE_FROM_RULE_SET(
```

```
  rule_set_name => 'Limit_DBA_Access',
  rule_name     => 'Check UPDATE operations');
END;
/
```

15.1.6 DELETE_RULE_SET Procedure

The `DELETE_RULE_SET` procedure deletes a rule set.

Syntax

```
DBMS_MACADM.DELETE_RULE_SET(
 rule_set_name IN VARCHAR2);
```

Parameters

Table 15-6 DELETE_RULE_SET Parameter

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view.

Example

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

15.1.7 RENAME_RULE Procedure

The `RENAME_RULE` procedure renames a rule and causes the name change to take effect everywhere the rule is used

Syntax

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

Parameters

Table 15-7 RENAME_RULE Parameters

Parameter	Description
`rule_name`	Current rule name. To find existing rules in the current database instance, query the `DBA_DV_RULE` view. To find rules that have been associated with rule sets, query `DBA_DV_RULE_SET_RULE`.
`new_name`	New rule name, up to 128 characters in mixed-case.

Table 15-7 (Cont.) RENAME_RULE Parameters

Parameter	Description
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule is in the application root

Example

```
BEGIN
 DBMS_MACADM.RENAME_RULE(
  rule_name  => 'Check UPDATE operations',
  new_name   => 'Check Sector 2 Processes');
END;
/
```

15.1.8 RENAME_RULE_SET Procedure

The RENAME_RULE_SET procedure renames a rule set and causes the name change to take effect everywhere the rule set is used.

Syntax

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

Parameters

Table 15-8 RENAME_RULE_SET Parameters

Parameter	Description
rule_set_name	Current rule set name. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view.
new_name	New rule set name, up to 128 characters in mixed-case. Spaces are allowed.
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the rule set is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the rule set is in the application root

Example

```
BEGIN
 DBMS_MACADM.RENAME_RULE_SET(
  rule_set_name => 'Limit_DBA_Access',
  new_name      => 'Limit Sector 2 Access');
```

```
END;
/
```

15.1.9 UPDATE_RULE Procedure

The `UPDATE_RULE` procedure updates a rule.

Syntax

```
DBMS_MACADM.UPDATE_RULE(
 rule_name  IN VARCHAR2,
 rule_expr  IN VARCHAR2,
 is_static  IN BOOLEAN DEFAULT FALSE);
```

Parameters

Table 15-9 UPDATE_RULE Parameters

Parameter	Description
`rule_name`	Rule name. To find existing rules in the current database instance. To find rules that have been associated with rule sets, query `DBA_DV_RULE_SET_RULE`.
`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 Rule for more information on rule expressions. To find existing rule expressions, query the `DBA_DV_RULE` view.
`is_static`	Specifies whether the evaluation of a rule should be static or dynamic. When a rule is specified as static, then the result of the first rule evaluation in a session will be cached and used whenever the rule needs to be evaluated again in the session.

Example

```
BEGIN
DBMS_MACADM.UPDATE_RULE(
  rule_name  => 'Check UPDATE operations',
  rule_expr  =>'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''SYSADM'' AND
               (
                 UPPER(SYS_CONTEXT(''USERENV'',''MODULE'')) LIKE ''APPSRVR%'' OR
                 UPPER(SYS_CONTEXT(''USERENV'',''MODULE'')) LIKE ''DBAPP%''
               )',
  is_static  => TRUE);
END;
/
```

15.1.10 UPDATE_RULE_SET Procedure

The `UPDATE_RULE_SET` procedure updates a rule set.

When you update a rule set, the existing traditional audit records are disabled. You must create a unified audit policy to capture new audit records.

Syntax

```
DBMS_MACADM.UPDATE_RULE_SET(
 rule_set_name    IN VARCHAR2,
 description      IN VARCHAR2,
 enabled          IN VARCHAR2,
 eval_options     IN NUMBER,
 audit_options    IN NUMBER,
 fail_options     IN NUMBER,
 fail_message     IN VARCHAR2,
 fail_code        IN NUMBER,
 handler_options  IN NUMBER,
 handler          IN VARCHAR2,
 is_static        IN BOOLEAN DEFAULT);
```

Parameters

Table 15-10 UPDATE_RULE_SET Parameters

Parameter	Description
`rule_set_name`	Rule set name. To find existing rule sets in the current database instance.
`description`	Description of the purpose of the rule set, up to 1024 characters in mixed-case.
`enabled`	`DBMS_MACUTL.G_YES` (Yes) enables rule set checking; `DBMS_MACUTL.G_NO` (No) disables it. The default for the `enabled` setting is the previously set value, which you can find by querying the `DBA_DV_RULE_SET` data dictionary view.
`eval_options`	If you plan to assign multiple rules to the rule set, enter one of the following settings: • `DBMS_MACUTL.G_RULESET_EVAL_ALL`: All rules in the rule set must evaluate to true for the rule set itself to evaluate to true. • `DBMS_MACUTL.G_RULESET_EVAL_ANY`: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to true. The default for `eval_options` is the previously set value, which you can find by querying the `DBA_DV_RULE_SET` data dictionary view.

Table 15-10 (Cont.) UPDATE_RULE_SET Parameters

Parameter	Description
audit_options	Perform one of the following actions: • Set DBMS_MACUTL.G_RULESET_AUDIT_OFF to disable traditional auditing for this rule set policy. • Leave the current existing audit_options setting as is if you want to continue using this traditional auditing setting. To do so, you can specify NULL, or specify the same setting as the current existing audit_options for this rule set, or simply do not specify audit_options (the default is NULL). Starting with Oracle Database 23c, traditional auditing is desupported. To audit a rule set, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a rule set.
fail_options	Options for reporting errors: • DBMS_MACUTL.G_RULESET_FAIL_SHOW: Shows an error message. • DBMS_MACUTL.G_RULESET_FAIL_SILENT: Does not show an error message. The default for fail_options is the previously set value, which you can find by querying the DBA_DV_RULE_SET data dictionary view.
fail_message	Error message for failure, up to 80 characters in mixed-case, to associate with the fail code you specify for fail_code.
fail_code	Enter a number in the range of -20000 to -20999 or 20000 to 20999 to associate with the fail_message parameter.
handler_options	Select one of the following settings: • DBMS_MACUTL.G_RULESET_HANDLER_OFF: Disables error handling. • DBMS_MACUTL.G_RULESET_HANDLER_FAIL: Call handler on rule set failure. • DBMS_MACUTL.G_RULESET_HANDLER_SUCCESS: Call handler on rule set success. The default for handler_options is the previously set value, which you can find by querying the DBA_DV_RULE_SET data dictionary view.
handler	Name of the PL/SQL function or procedure that defines the custom event handler logic.
is_static	Optional. Determines how often a rule set is evaluated when it is accessed by a SQL statement. The default is FALSE. • TRUE: The rule set is evaluated once during the user session. After that, the value is re-used. • FALSE: The rule set evaluated each time a SQL statement accesses it.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ANY,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
```

```
  fail_message     => 'Access denied!',
  fail_code        => 20900,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_OFF,
  handler          => '',
  is_static        => TRUE);
END;
/
```

If you did not make any modifications to the `audit_options` parameter of your rule set, then the existing traditional audit policy will still be in place. If you updated the `audit_options` parameter of the rule set, then this auditing will be disabled because traditional auditing is desupported starting in release 23c. To capture new audit records, create and enable a unified audit policy. For example:

```
CREATE AUDIT POLICY Limit_DBA_Access_pol
 ACTIONS SELECT, UPDATE, DELETE
 ACTIONS COMPONENT=DV RULE SET FAILURE ON "Limit_DBA_Access";

AUDIT POLICY Limit_DBA_Access_pol EXCEPT psmith;
```

You can view the audit records by querying the `UNIFIED_AUDIT_TRAIL` data dictionary view. See *Oracle Database Security Guide* for how this works.

Related Topics

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

15.2 Oracle Database Vault PL/SQL Rule Set Functions

Oracle Database Vault provides functions to use in rule sets to inspect the SQL statement that the rule set protects.

- DV_SYSEVENT Function
  The `DV_SYSEVENT` function returns the system event firing the rule set. .
- DV_LOGIN_USER Function
  The `DV_LOGIN_USER` function returns the session user name, in `VARCHAR2` data type.
- DV_INSTANCE_NUM Function
  The `DV_INSTANCE_NUM` function returns the database instance number, in `NUMBER` data type.
- DV_DATABASE_NAME Function
  The `DV_DATABASE_NAME` function returns the database name, in `VARCHAR2` data type.
- DV_DICT_OBJ_TYPE Function
  The `DV_DICT_OBJ_TYPE` function returns the type of the dictionary object on which the database operation occurred.
- DV_DICT_OBJ_OWNER Function
  The `DV_DICT_OBJ_OWNER` function returns the name of the owner of the dictionary object on which the database operation occurred.

- DV_DICT_OBJ_NAME Function
  The `DV_DICT_OBJ_NAME` function returns the name of the dictionary object on which the database operation occurred.
- DV_SQL_TEXT Function
  The `DV_SQL_TEXT` function returns the first 4000 characters of SQL text of the database statement used in the operation.

15.2.1 DV_SYSEVENT Function

The `DV_SYSEVENT` function returns the system event firing the rule set. .

The event name is the same as that in the syntax of the SQL statement (for example, `INSERT`, `CREATE`.) The return type is `VARCHAR2`.

Syntax

```
DV_SYSEVENT ()
RETURN VARCHAR2;
```

Parameters

None

Example

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

15.2.2 DV_LOGIN_USER Function

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

Syntax

```
DV_LOGIN_USER ()
RETURN VARCHAR2;
```

Parameters

None

Example

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

15.2.3 DV_INSTANCE_NUM Function

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

Syntax

```
DV_INSTANCE_NUM ()
RETURN NUMBER;
```

Parameters

None

Example

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

15.2.4 DV_DATABASE_NAME Function

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

Syntax

```
DV_DATABASE_NAME ()
RETURN VARCHAR2;
```

Parameters

None

Example

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

15.2.5 DV_DICT_OBJ_TYPE Function

The `DV_DICT_OBJ_TYPE` function returns the type of the dictionary object on which the database operation occurred.

For example, dictionary objects it returns are table, procedure, or view. The return type is `VARCHAR2`.

Syntax

```
DV_DICT_OBJ_TYPE ()
RETURN VARCHAR2;
```

Parameters

None

Example

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

15.2.6 DV_DICT_OBJ_OWNER Function

The `DV_DICT_OBJ_OWNER` function returns the name of the owner of the dictionary object on which the database operation occurred.

The return type is `VARCHAR2`.

Syntax

```
DV_DICT_OBJ_OWNER ()
RETURN VARCHAR2;
```

Parameters

None

Example

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

15.2.7 DV_DICT_OBJ_NAME Function

The `DV_DICT_OBJ_NAME` function returns the name of the dictionary object on which the database operation occurred.

The return type is `VARCHAR2`.

Syntax

```
DV_DICT_OBJ_NAME ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
```

```
  rule_name => 'Check Dictionary Object Name',
  rule_expr => 'DV_DICT_OBJ_NAME = ''SALES''');
END;
/
```

15.2.8 DV_SQL_TEXT Function

The `DV_SQL_TEXT` function returns the first 4000 characters of SQL text of the database statement used in the operation.

The return type is `VARCHAR2`.

Syntax

```
DV_SQL_TEXT ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check SQL Text',
  rule_expr => 'DV_SQL_TEXT = ''SELECT SALARY FROM HR.EMPLOYEES''');
END;
/
```

16

Oracle Database Vault Command Rule APIs

The DBMS_MACADM PL/SQL package provides procedures for configuring command rules. .

Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures.

- CREATE_COMMAND_RULE Procedure
  The CREATE_COMMAND_RULE procedure creates both command and local command rules, which can be added to a rule set.
- CREATE_CONNECT_COMMAND_RULE Procedure
  The CREATE_CONNECT_COMMAND_RULE procedure creates both common and local CONNECT command rules that you can associate with a user and a rule set.
- CREATE_SESSION_EVENT_CMD_RULE Procedure
  The CREATE_SESSION_EVENT_CMD_RULE procedure creates both common and local command rules that you can associate with session events, based on the ALTER SESSION statement.
- CREATE_SYSTEM_EVENT_CMD_RULE Procedure
  The CREATE_SYSTEM_EVENT_CMD_RULE procedure creates both command and local command rules that you can associate with system events, based on the ALTER SYSTEM statement.
- DELETE_COMMAND_RULE Procedure
  The DELETE_COMMAND_RULE procedure drops a command rule declaration.
- DELETE_CONNECT_COMMAND_RULE Procedure
  The DELETE_CONNECT_COMMAND_RULE procedure deletes a CONNECT command rule that had been created with the CREATE_CONNECT_COMMAND_RULE procedure.
- DELETE_SESSION_EVENT_CMD_RULE Procedure
  The DELETE_SESSION_EVENT_CMD_RULE procedure deletes a session command rule that was associated with events.
- DELETE_SYSTEM_EVENT_CMD_RULE Procedure
  The DELETE_SYSTEM_EVENT_CMD_RULE procedure deletes a system command rule that was associated with events.
- UPDATE_COMMAND_RULE Procedure
  The UPDATE_COMMAND_RULE procedure updates the command rule declaration for both common and local command rules.
- UPDATE_CONNECT_COMMAND_RULE Procedure
  The UPDATE_CONNECT_COMMAND_RULE procedure updates a CONNECT command rule that had been created with the CREATE_CONNECT_COMMAND_RULE procedure.
- UPDATE_SESSION_EVENT_CMD_RULE Procedure
  The UPDATE_SESSION_EVENT_CMD_RULE procedure updates both common and local session event command rules, based on the ALTER SESSION statement.
- UPDATE_SYSTEM_EVENT_CMD_RULE Procedure
  The UPDATE_SYSTEM_EVENT_CMD_RULE procedure updates both common and local system event command rules, based on the ALTER SYSTEM statement.

Related Topics

- Configuring Command Rules
  You can create command rules or use the default command rules to protect DDL and DML statements.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the DBMS_MACUTL PL/SQL package.

16.1 CREATE_COMMAND_RULE Procedure

The CREATE_COMMAND_RULE procedure creates both command and local command rules, which can be added to a rule set.

Optionally, you can use it to enable the command rule for rule checking with a rule set.

Syntax

```
DBMS_MACADM.CREATE_COMMAND_RULE(
 command          IN VARCHAR2,
 rule_set_name    IN VARCHAR2,
 object_owner     IN VARCHAR2,
 object_name      IN VARCHAR2,
 enabled          IN VARCHAR2,
 privilege_scope  IN NUMBER,
 clause_name      IN VARCHAR2,
 parameter_name   IN VARCHAR2,
 event_name       IN VARCHAR2,
 component_name   IN VARCHAR2,
 action_name      IN VARCHAR2,
 scope            IN NUMBER DEFAULT);
```

Parameters

Table 16-1 CREATE_COMMAND_RULE Parameters

Parameter	Description
command	SQL statement to protect. To find existing command rules, query the DBA_DV_COMMAND_RULE data dictionary view. See also Related Topics.
rule_set_name	Name of rule set to associate with this command rule. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view.
object_owner	Database schema to which this command rule will apply. The wildcard % is allowed, except for the SELECT, INSERT, UPDATE, DELETE, and EXECUTE statements. To find the available users, query the DBA_USERS view.
object_name	Object to be protected by the command rule. (The wildcard % is allowed.) To find the available objects, query the ALL_OBJECTS view.

Table 16-1 (Cont.) CREATE_COMMAND_RULE Parameters

Parameter	Description
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' (Yes) to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
privilege_scope	Obsolete parameter
clause_name	A clause from the SQL statement that was used to create the command rule. For example, a command rule for the ALTER SESSION SQL statement could have the SET clause as the clause_name parameter. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
parameter_name	A parameter from the clause_name parameter. For example, for an ALTER SESSION command rule, you could set parameter_name to EVENTS if the clause_name is SET. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
event_name	An event that the command rule defines. For example, suppose an ALTER SESSION command rule uses SET for the clause_name and EVENTS as the parameter_name. The event_name could be set to TRACE if you want to track trace events. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to EVENTS.
component_name	A component of the event_name setting. For example, for a TRACE event, the component_name could be GCS. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to EVENTS.
action_name	An action of the component_name setting. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to EVENTS.
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you create the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`

ALTER SYSTEM Command Rule Settings

Table 16-2 describes the ALTER SYSTEM command rule settings.

Table 16-2 ALTER SYSTEM Command Rule Settings

clause_name	parameter_name — Parameter Value
ARCHIVE LOG	• ALL — *sequence_number* • CHANGE — *change_number* • CURRENT — N/A • GROUP — *group_number* • LOGFILE — *log_file_name* • NEXT — N/A • SEQUENCE — N/A
CHECK DATAFILES	N/A — global or local
CHECKPOINT	N/A — global or local
COPY LOGFILE	N/A — N/A
DISTRIBUTED RECOVERY	N/A — enable or disable
DUMP	• DATAFILE — N/A • FLASHBACK — N/A • LOGFILE — N/A • REDO — N/A • TEMPFILE — N/A • UNDO — N/A
END SESSION	DISCONNECT SESSION — N/A KILL SESSION — N/A
FLUSH	BUFFER_CACHE — N/A GLOBAL CONTEXT — N/A REDO — *target_db_name* SHARED_POOL — N/A
QUIESCE	QUIESCE RESTRICTED — N/A UNQUIESCE — N/A
REFRESH	LDAP_REGISTRATION — N/A
REGISTER	N/A — N/A
RESET	*initialization_parameter_name* — N/A
RESUME	N/A — N/A
SECURITY	RESTRICTED SESSION — enable or disable SET ENCRYPTION KEY — N/A SET ENCRYPTION WALLET — open or close
SET	EVENTS — *event_string* GLOBAL_TOPIC_ENABLED — true or false *initialization_parameter_name* — *parameter_value* LDAP_REGISTRATION_ENABLED — true or false LDAP_REG-SYNC_INTERVAL — Number SINGLETASK DEBUG — N/A USE_STORED_OUTLINES — true , false, or *category_name*

Table 16-2 (Cont.) ALTER SYSTEM Command Rule Settings

clause_name	parameter_name — Parameter Value
SHUTDOWN DISPPATCHER	N/A — *dispatcher_name*
SWITCH LOGFILE	N/A — all or none
SUSPEND	N/A — N/A
TX RECOVERY	N/A — enable or disable

ALTER SESSION Command Rule Settings

Table 16-3 describes the ALTER SESSION command rule settings.

Table 16-3 ALTER SESSION Command Rule Settings

clause_name	parameter_name — Parameter Value
ADVISE	N/A — COMMIT, ROLLBACK, or NOTHING
CLOSE DATABASE LINK	N/A — *database_link*
COMMIT IN PROCEDURE	N/A — ENABLE or DISABLE
GUARD	N/A — ENABLE or DISABLE
ILM	ROW ACCESS TRACKING — N/A ROW MODIFICATION TRACKING — N/A
LOGICAL REPLICATION	N/A — N/A
PARALLEL DML	N/A — ENABLE, DISABLE, or FORCE
PARALLEL DDL	N/A — ENABLE, DISABLE, or FORCE
PARALLEL QUERY	N/A — ENABLE, DISABLE, or FORCE
RESUMABLE	N/A — ENABLE or DISABLE
SYNC WITH PRIMARY	N/A — N/A

Table 16-3 (Cont.) ALTER SESSION Command Rule Settings

clause_name	parameter_name — Parameter Value
SET	APPLICATION ACTION — *action_name* APPLICATION MODULE — *module_name* CONSTRAINTS — IMMEDIATE, DEFERRED, or DEFAULT CONTAINER — *container_name* CURRENT SCHEMA — *schema_name* EDITION — *edition_name* ERROR ON OVERLAP TIME — TRUE or FALSE EVENTS — *event_string* FLAGGER — OFF, FULL, INTERMEDIATE, ENTRY *initialization_parameter_name* — *parameter_name* INSTANCE — *instance_number* ISOLATION_LEVEL — SERIALIZABLE or READ COMMITTED ROW_ARCHIVAL_VISABILITY — ACTIVE or ALL SQL_TRANSFORMATION_PROFILE — *profile_name* STANDBY_MAX_DATA_DELAY — NONE*number* TIME_ZONE — LOCAL, DBTIMEZONE, or *other_value* USE_PRIVATE_OUTLINES — TRUE, FALSE, or *category_name* USE_STORED_OUTLINES — TRUE, FALSE, or *category_name*

Examples

Simple Command Rules

The following example shows how to create a simple command rule for the `SELECT` statement on the `HR.EMPLOYEEES` table. This command rule uses a custom rule set called `Check User Role`. This rule set must exist before the command rule can be created.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'SELECT',
  rule_set_name   => 'Check User Role',
  object_owner    => 'HR,
  object_name     => 'EMPLOYEES,
  enabled         => DBMS_MACUTL.G_YES);
END;
/
```

This example shows how to create a command rule that checks if users can enable or disable the `hr_app_aud_pol` unified audit policy. Note that if the object is a unified audit policy, then you must have `AUDIT POLICY`, not `AUDIT`, for the `command` parameter.

```
BEGIN
DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'AUDIT POLICY',
  rule_set_name   => 'Check ability to audit',
  object_owner    => '%',
  object_name     => 'hr_app_aud_pol',
  enabled         => DBMS_MACUTL.G_YES,
```

```
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

ALTER SESSION Command Rule Using the SET Clause

The following example shows how to create an `ALTER SESSION` command rule that uses the `SET` clause with the `ERROR_ON_OVERLAP_TIME` parameter.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'ALTER SESSION',
  rule_set_name   => 'Test ERROR_ON_OVERLAP_TIME for FALSE',
  object_owner    => '%',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES,
  clause_name     => 'SET',
  parameter_name  => 'ERROR_ON_OVERLAP_TIME',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

In this example:

- `rule_set_name`: The `ALTER SESSION` SQL statement `ERROR_ON_OVERLAP_TIME` session parameter must be set to either `TRUE` or `FALSE`. You can create a rule set that checks if this setting. For example, for the rule:

  ```
  EXEC DBMS_MACADM.CREATE_RULE('RULE_TRUE', 'UPPER(PARAMETER_VALUE) = ''TRUE''');
  ```

  The rule set that is used with this rule can be similar to the following:

  ```
  BEGIN
   DBMS_MACADM.CREATE_RULE_SET(
    rule_set_name    => 'Test ERROR_ON_OVERLAP_TIME',
    description      => 'Checks if the ERROR_ON_OVERLAP_TIME setting is TRUE or
  FALSE',
    enabled          => DBMS_MACUTL.G_YES,
    eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
    audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
    fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SILENT,
    fail_message     => 'false error on overlaptime',
    fail_code        => 20461,
    handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
    handler          => '',
    is_static        => false);
  END;
  /
  EXEC DBMS_MACADM.ADD_RULE_TO_RULE_SET('Test ERROR_ON_OVERLAP_TIME', 'RULE_TRUE');
  ```

- `object_owner` and `object_name` must be set to `%` for `ALTER SESSION` and `ALTER SYSTEM` command rules.
- `enabled` uses the `DBMS_MACUTL.G_YES` constant to enable the command rule when it is created.
- `clause_name` sets the `ALTER SESSION` command rule to use the `SET` clause of the `ALTER SESSION` PL/SQL statement.
- `parameter_name` is set to the `ERROR_ON_OVERLAP_TIME` parameter of the `SET` clause.

- `scope` uses the `DBMS_MACUTL.G_SCOPE_COMMON` constant to set the command rule to be a common command rule. This command rule will be in the application root of a multitenant environment, so the user running this procedure must be in the CDB root. Any rules or rule sets that are associated with this command rule must be common.

  If you were creating the command rule locally, you would set `scope` to `DBMS_MACUTL.G_SCOPE_LOCAL`. In that case, the user who runs this procedure must be in the PDB in which the command rule will reside. To find the existing PDBs, you can query the `DBA_PDBS` data dictionary view. Any rules or rule sets that are associated with this command rule must be local.

ALTER SYSTEM Command Rule Using the CHECKPOINT Clause

This example shows how to create an ALTER SYSTEM command rule that users the `CHECKPOINT` clause. To have the command rule test for the `CHECKPOINT` setting, you must create a rule set and rule, similar to the ALTER SESSION command rule in the previous example. In this example, the `parameter` setting is not specified because the `CHECKPOINT` setting does not have parameters.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'ALTER SYSTEM',
  rule_set_name   => 'Test CHECKPOINT Setting',
  object_owner    => '%',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES,
 clause_name     => 'CHECKPOINT',
 parameter_name  => '',
 scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

ALTER SESSION Command Rule Using the SET Clause

The following ALTER SESSION command rule uses the `SET` clause to specify an `event_name` and `component_name`. You can only use the `event_name`, `component_name`, and `action_name` parameters if the `clause_name` parameter specifies `SET`.

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'ALTER SESSION',
  rule_set_name   => 'Check Trace Events',
  object_owner    => '%',
  object_name     => '%',
  enabled         => DBMS_MACUTL.G_YES,
  clause_name     => 'SET',
  parameter_name  => 'EVENTS',
  event_name      => 'TRACE',
  component_name  => 'GCS',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

Related Topics

- ALTER SESSION and ALTER SYSTEM Command Rules
  You can create different kinds of `ALTER SESSION` and `ALTER SYSTEM` command rules that provide fine-grained control for these SQL statements.

- SQL Statements That Can Be Protected by Command Rules
  You can protect a large number of SQL statements by using command rules.

16.2 CREATE_CONNECT_COMMAND_RULE Procedure

The `CREATE_CONNECT_COMMAND_RULE` procedure creates both common and local CONNECT command rules that you can associate with a user and a rule set.

Syntax

```
DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
 user_name       IN VARCHAR2,
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 16-4 CREATE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
`user_name`	User to whom the CONNECT command rule will apply. If you enter the `%` wildcard, then the CONNECT command rule will be applied to every database user. If you run this procedure in the root, then specifying `%` applies to all common users. If you run the procedure in a PDB, then it applies to all local and common users who have access to this PDB. If there are two command rules, one common and one local, and they both apply to the same object, then both must evaluate successfully for the operation to succeed. Ensure that this user is common if the CONNECT command rule is common, and local or common if the CONNECT command rule is local. To find existing database users in the current instance, query the `DBA_USERS` view, described in *Oracle Database Reference*.
`rule_set_name`	Name of rule set to associate with this command rule. Ensure that this rule set is common if the CONNECT command rule is common, and local if the CONNECT command rule is local. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view, described in DBA_DV_RULE_SET View.
`enabled`	Specify one of the following options to set the status of the command rule: • `DBMS_MACUTL.G_YES` or `'y'` (Yes) to enable the command rule (default) • `DBMS_MACUTL.G_NO` or `'n'` to disable the command rule, including the capture of violations in the simulation log • `DBMS_MACUTL.G_SIMULATION` or `'s'` to enable SQL statements to execute but capture violations in the simulation log

Table 16-4 (Cont.) CREATE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you create the common CONNECT command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`

Examples

The following example shows how to create a common CONNECT command rule. This command rule will be in the CDB root, so the user who runs this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
  rule_set_name   => 'Allow Sessions',
  user_name       => 'C##HR_ADMIN',
  enabled         => DBMS_MACUTL.G_SIMULATION,
  scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This example is a local version of the preceding example. The user who runs this procedure must be in the PDB in which the local CONNECT command rule will reside. To find the available PDBs, run the `show pdbs` command. Any rule sets that are associated with this command rule must be local. The user can be either common or local.

```
BEGIN
 DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE(
 rule_set_name   => 'Allow Sessions',
 user_name       => 'PSMITH',
 enabled         => DBMS_MACUTL.G_SIMULATION,
 scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

16.3 CREATE_SESSION_EVENT_CMD_RULE Procedure

The CREATE_SESSION_EVENT_CMD_RULE procedure creates both common and local command rules that you can associate with session events, based on the ALTER SESSION statement.

Syntax

```
DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE(
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT,
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 16-5 CREATE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
rule_set_name	Name of the rule set to associate with the command rule. Ensure that this rule set is common if the session event command rule is common, and local if the command rule is local. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view.
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' (Yes) to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
event_name	An event that the command rule defines. This setting enables the command rule to correspond with an ALTER SESSION SET EVENTS *event_name* statement. For example, to track trace events, you would set *event_name* to TRACE.
component_name	A component of the event_name setting. Example settings are DV, OLS, or GCS. You can find valid component names by issuing ORADEBUG DOC COMPONENT RDBMS as user SYS. The output displays parent and child components, which you can use for the component_name setting. For example, both XS (parent) and XSSESSION (child of XS) are valid component names. If you select the parent component, then the command rule applies to it and the child components.
action_name	An action of the component_name setting

Table 16-5 (Cont.) CREATE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you create the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record. The default is FALSE.

Examples

The following example shows how to create a common session event command rule in a multitenant environment. This command rule will be in the application root, so the user running this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE(
  rule_set_name   => 'Allow Sessions',
  event_name      => 'TRACE',
  component_name  => 'DV',
  action_name     => 'CURSORTRACE',
  enabled         => DBMS_MACUTL.G_SIMULATION,
  scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

This example shows how to create a session event for the 47998 trace event. This example will records the PL/SQL stack for failed operations.

```
BEGIN
 DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE(
  rule_set_name   => 'Allow Sessions',
  event_name      => '47998',
  enabled         => 'y',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL,
  pl_sql_stack    => TRUE);
END;
/
```

16.4 CREATE_SYSTEM_EVENT_CMD_RULE Procedure

The `CREATE_SYSTEM_EVENT_CMD_RULE` procedure creates both command and local command rules that you can associate with system events, based on the `ALTER SYSTEM` statement.

Syntax

```
DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE(
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 16-6 CREATE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
`rule_set_name`	Name of the rule set to associate with the command rule. Ensure that this rule set is common if the system event command rule is common, and local if the command rule is local. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view.
`event_name`	An event that the command rule defines. This setting enables the command rule to correspond to an `ALTER SYSTEM SET EVENTS` *`event_name`* statement. For example, to track trace events, you would set *`event_name`* to `TRACE`.
`component_name`	A component of the `event_name` setting. Example settings are `DV`, `OLS`, or `GCS`. You can find valid component names by issuing `ORADEBUG DOC COMPONENT RDBMS` as user `SYS`. The output displays parent and child components, which you can use for the `component_name` setting. For example, both `XS` (parent) and `XSSESSION` (child of `XS`) are valid component names. If you select the parent component, then the command rule applies to it and the child components.
`action_name`	An action of the `component_name` setting
`enabled`	Specify one of the following options to set the status of the command rule: • `DBMS_MACUTL.G_YES` or `'y'` to enable the command rule (default) • `DBMS_MACUTL.G_NO` or `'n'` to disable the command rule, including the capture of violations in the simulation log • `DBMS_MACUTL.G_SIMULATION` or `'s'` to enable SQL statements to execute but capture violations in the simulation log

Table 16-6 (Cont.) CREATE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you create the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record. The default is FALSE.

Example

The following example shows how to create a common system event command rule in a multitenant environment. This command rule will be in the application root, so the user running this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE(
  rule_set_name   => 'Enabled',
  event_name      => 'TRACE',
  component_name  => 'GSIPC',
  action_name     => 'HEAPDUMP',
  enabled         => DBMS_MACUTL.G_YES,
  scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

16.5 DELETE_COMMAND_RULE Procedure

The DELETE_COMMAND_RULE procedure drops a command rule declaration.

Syntax

```
DBMS_MACADM.DELETE_COMMAND_RULE(
 command         IN VARCHAR2,
 object_owner    IN VARCHAR2,
 object_name     IN VARCHAR2,
 clause_name     IN VARCHAR2,
 parameter_name  IN VARCHAR2 DEFAULT,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 16-7 DELETE_COMMAND_RULE Parameters

Parameter	Description
command	SQL statement the command rule protects. To find available command rules, query the DBA_DV_COMMAND_RULE view.
object_owner	Database schema to which this command rule applies. To find the available users in the current database instance, query the DBA_USERS view.
object_name	Object name. The wildcard % is allowed. To find the available objects in the current database instance, query the ALL_OBJECTS view.
clause_name	A clause from the SQL statement that was used to create the command rule. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
parameter_name	A parameter from the clause_name parameter. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
event_name	An event that the command rule defines. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
component_name	A component of the event_name setting. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
action_name	An action of the component_name setting. Applies only to command rules for ALTER SYSTEM and ALTER SESSION.
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Examples

When you drop a command rule, you must omit the rule_set_name and enabled parameters, and ensure that the rest of the parameters match the settings that were used the last time the command rule was updated. You can check the most recent settings by querying the DBA_DV_COMMAND_RULE data dictionary view.

For example, suppose you created the following command rule:

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE(
  command         => 'SELECT',
  rule_set_name   => 'Enabled',
  object_owner    => 'OE',
  object_name     => 'ORDERS',
  enabled         => DBMS_MACUTL.G_YES,
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

To drop this command rule, use the most of same parameters as shown here, but omit `rule_set_name` and `enabled`.

```
BEGIN
 DBMS_MACADM.DELETE_COMMAND_RULE(
  command         => 'SELECT',
  object_owner    => 'OE',
  object_name     => 'ORDERS',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

The following example shows how to delete an `ALTER SESSION` command rule.

```
BEGIN
 DBMS_MACADM.DELETE_COMMAND_RULE(
  command         => 'ALTER SESSION',
  object_owner    => '%',
  object_name     => '%',
  clause_name     => 'SET',
  parameter_name  => 'EVENTS',
  event_name      => 'TRACE',
  component_name  => 'GCS',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

Related Topics

- DBA_DV_COMMAND_RULE View
  The `DBA_DV_COMMAND_RULE` data dictionary view lists the SQL statements that are protected by command rules.

16.6 DELETE_CONNECT_COMMAND_RULE Procedure

The `DELETE_CONNECT_COMMAND_RULE` procedure deletes a `CONNECT` command rule that had been created with the `CREATE_CONNECT_COMMAND_RULE` procedure.

Syntax

```
DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE(
 user_name       IN VARCHAR2,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 16-8 DELETE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
`user_name`	User to whom the `CONNECT` command rule applied. To find this user, query the `OBJECT_OWNER` field of the `DBA_DV_COMMAND_RULE` view.

Table 16-8 (Cont.) DELETE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Example

```
BEGIN
 DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE(
  user_name       => 'PSMITH',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

16.7 DELETE_SESSION_EVENT_CMD_RULE Procedure

The DELETE_SESSION_EVENT_CMD_RULE procedure deletes a session command rule that was associated with events.

Syntax

```
DBMS_MACADM.DELETE_SESSION_EVENT_CMD_RULE(
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 16-9 DELETE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
event_name	An event that the session event command rule defines. The DBA_CV_COMMAND_RULE view lists information about existing command rules.
component_name	A component of the event_name setting
action_name	An action of the component_name setting
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Example

The following example shows how to delete a common session event command rule in the application root a multitenant environment. The user running this procedure must be a

common user in the CDB root. When you specify the parameters, ensure that they match exactly the parameters that were used the last time the command rule was updated. To find the current settings of the command rule, query the DBA_DV_COMMAND_RULE view.

```
BEGIN
DBMS_MACADM.DELETE_SESSION_EVENT_CMD_RULE(
 event_name      => '47999',
 scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
 /
```

16.8 DELETE_SYSTEM_EVENT_CMD_RULE Procedure

The DELETE_SYSTEM_EVENT_CMD_RULE procedure deletes a system command rule that was associated with events.

Syntax

```
DBMS_MACADM.DELETE_SYSTEM_EVENT_CMD_RULE(
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 16-10 DELETE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
event_name	An event that the system event command rule defines. The DBA_DV_COMMAND_RULE view lists information about existing command rules.
component_name	A component of the event_name setting
action_name	An action of the component_name setting
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Examples

The following example shows how to delete a common system event command rule in the application root. The user running this procedure must be a common user in the CDB root. When you specify the parameters, ensure that they match exactly the parameters that were used the last time the command rule was updated. To find the current settings of the command rule, query the DBA_DV_COMMAND_RULE view.

```
BEGIN
 DBMS_MACADM.DELETE_SYSTEM_EVENT_CMD_RULE(
 event_name      => 'TRACE',
 component_name  => 'DV',
 action_name     => '',
```

```
 scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

16.9 UPDATE_COMMAND_RULE Procedure

The `UPDATE_COMMAND_RULE` procedure updates the command rule declaration for both common and local command rules.

Syntax

```
DBMS_MACADM.UPDATE_COMMAND_RULE(
 command         IN VARCHAR2,
 rule_set_name   IN VARCHAR2,
 object_owner    IN VARCHAR2,
 object_name     IN VARCHAR2,
 enabled         IN VARCHAR2,
 privilege_scope IN NUMBER,
 clause_name     IN VARCHAR2,
 parameter_name  IN VARCHAR2 DEFAULT,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT,
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 16-11 UPDATE_COMMAND_RULE Parameters

Parameter	Description
`command`	Command rule to update See also Related Topics.
`rule_set_name`	Name of rule set to associate with this command rule. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view.
`object_owner`	Database schema to which this command rule applies. To find the available users, query the `DBA_USERS` view. See also Related Topic on creating a command rule for more details about object owners.
`object_name`	Object name. (The wildcard % is allowed. See also Related Topic on creating a command rule for more details about object names. To find the available objects, query the `ALL_OBJECTS` view.
`enabled`	Specify one of the following options to set the status of the command rule: • `DBMS_MACUTL.G_YES` or `'y'` to enable the command rule (default) • `DBMS_MACUTL.G_NO` or `'n'` to disable the command rule, including the capture of violations in the simulation log • `DBMS_MACUTL.G_SIMULATION` or `'s'` to enable SQL statements to execute but capture violations in the simulation log
`privilege_scope`	Obsolete parameter

Table 16-11 (Cont.) UPDATE_COMMAND_RULE Parameters

Parameter	Description
clause_name	A clause from the SQL statement that was used to create the command rule. For example, a command rule for the ALTER SESSION SQL statement could have the SET clause as the clause_name parameter. Applies only to command rules for ALTER SYSTEM and ALTER SESSION. The command rule settings for these two statements are described in the DBMS_MACADM.CREATE_COMMAND_RULE procedure. See Related Topics.
parameter_name	A parameter from the clause_name parameter. For example, for an ALTER SESSION command rule, you could set parameter_name to EVENTS if the clause_name is SET. Applies only to command rules for ALTER SYSTEM and ALTER SESSION. See Related Topics.
event_name	An event that the command rule defines. For example, for an ALTER SESSION command rule that uses SET for the clause_name and EVENTS as the parameter_name, then the event_name could be set to TRACE. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to events. See Related Topics.
component_name	A component of the event_name setting. For example, for a TRACE event, the component_name could be GCS. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to events. See Related Topics.
action_name	An action of the component_name setting. For example, if component_name is set to GCS, then the action_name setting could be DISK HIGH. Applies only to ALTER SYSTEM and ALTER SESSION command rules that have the parameter parameter set to events. See Related Topics.
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you update the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Examples

The following example shows how to update a simple command rule that protects the HR.EMPLOYEES schema (for example, changing its rule set).

```
BEGIN
 DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'SELECT',
  rule_set_name   => 'Disabled',
```

```
  object_owner    => 'HR',
  object_name     => 'EMPLOYEES',
  enabled         => DBMS_MACUTL.G_SIMULATION,
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

This example shows how to update a more complex command rule, which is based on the `ALTER SESSION` SQL statement.

```
BEGIN
 DBMS_MACADM.UPDATE_COMMAND_RULE(
  command         => 'ALTER SESSION',
  rule_set_name   => 'Enabled',
  object_owner    => '%',
  object_name     => '%',
  enabled         => 's',
  clause_name     => 'SET',
  parameter_name  => 'EVENTS',
  event_name      => 'TRACE',
  component_name  => 'GCS',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
END;
/
```

Related Topics

- SQL Statements That Can Be Protected by Command Rules
  You can protect a large number of SQL statements by using command rules.
- CREATE_COMMAND_RULE Procedure
  The `CREATE_COMMAND_RULE` procedure creates both command and local command rules, which can be added to a rule set.
- Creating a Command Rule
  You can create a different types of command rules using different command rule APIs.

16.10 UPDATE_CONNECT_COMMAND_RULE Procedure

The `UPDATE_CONNECT_COMMAND_RULE` procedure updates a `CONNECT` command rule that had been created with the `CREATE_CONNECT_COMMAND_RULE` procedure.

Syntax

```
DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE(
 user_name       IN VARCHAR2,
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 scope           IN NUMBER DEFAULT);
```

Parameters

Table 16-12 UPDATE_CONNECT_COMMAND_RULE Parameters

Parameter	Description
user_name	User to whom the CONNECT command rule will apply. If you enter the % wildcard, then the CONNECT command rule will be applied to every database user. If you run this procedure in the root, then specifying % applies to all common users. If you run the procedure in a PDB, then it applies to all local and common users who have access to this PDB. If there are two command rules, one common and one local, and they both apply to the same object, then both must evaluate successfully for the operation to succeed. Environment, ensure that this user is common if the CONNECT command rule is common, and local or common if the CONNECT command rule is local. To find existing command rules, query the DBA_DV_COMMAND_RULE view, described in DBA_DV_COMMAND_RULE View. To find existing database users in the current instance, query the DBA_USERS view, described in *Oracle Database Reference*.
rule_set_name	Name of rule set to associate with this command rule. Ensure that this rule set is common if the CONNECT command rule is common, and local if the CONNECT command rule is local. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View.
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you update the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`

Example

```
BEGIN
 DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE(
  rule_set_name   => 'Allow Sessions',
  user_name       => 'PSMITH',
  enabled         => 'DBMS_MACUTL.G_YES',
  scope           => DBMS_MACUTL.G_SCOPE_LOCAL);
```

```
END;
/
```

16.11 UPDATE_SESSION_EVENT_CMD_RULE Procedure

The `UPDATE_SESSION_EVENT_CMD_RULE` procedure updates both common and local session event command rules, based on the `ALTER SESSION` statement.

Syntax

```
DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE(
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT,
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 16-13 UPDATE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
`rule_set_name`	Name of the rule set to associate with the command rule. Ensure that this rule set is common if the session event command rule is common, and local if the command rule is local. To find existing rule sets in the current database instance, query the `DBA_DV_RULE_SET` view.
`enabled`	Specify one of the following options to set the status of the command rule: • `DBMS_MACUTL.G_YES` or `'y'` to enable the command rule (default) • `DBMS_MACUTL.G_NO` or `'n'` to disable the command rule, including the capture of violations in the simulation log • `DBMS_MACUTL.G_SIMULATION` or `'s'` to enable SQL statements to execute but capture violations in the simulation log
`event_name`	An event that the command rule defines. This setting enables the command rule to correspond with an `ALTER SESSION SET EVENTS` *`event_name`* statement. For example, to track trace events, you would set *`event_name`* to `TRACE`.
`component_name`	A component of the `event_name` setting. Example settings are `DV`, `OLS`, or `GCS`. You can find valid component names by issuing `ORADEBUG DOC COMPONENT RDBMS` as user `SYS`. The output displays parent and child components, which you can use for the `component_name` setting. For example, both `XS` (parent) and `XSSESSION` (child of `XS`) are valid component names. If you select the parent component, then the command rule applies to it and the child components.
`action_name`	An action of the `component_name` setting

Table 16-13 (Cont.) UPDATE_SESSION_EVENT_CMD_RULE Parameters

Parameter	Description
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you update the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Example

The following example shows how to update a common session event command rule. This command rule is in the application root, so the user running this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE(
  rule_set_name => 'Allow Sessions',
  event_name    => '47999',
  enabled       => DBMS_MACUTL.G_NO,
  scope         => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

16.12 UPDATE_SYSTEM_EVENT_CMD_RULE Procedure

The UPDATE_SYSTEM_EVENT_CMD_RULE procedure updates both common and local system event command rules, based on the ALTER SYSTEM statement.

Syntax

```
DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE(
 rule_set_name   IN VARCHAR2,
 enabled         IN VARCHAR2,
 event_name      IN VARCHAR2 DEFAULT,
 component_name  IN VARCHAR2 DEFAULT,
 action_name     IN VARCHAR2 DEFAULT,
 scope           IN NUMBER DEFAULT,
 pl_sql_stack    IN BOOLEAN DEFAULT);
```

Parameters

Table 16-14 UPDATE_SYSTEM_EVENT_CMD_RULE Parameters

Parameter	Description
rule_set_name	Name of the rule set to associate with the command rule. Ensure that this rule set is common if the system event command rule is common, and local if the command rule is local. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view.
enabled	Specify one of the following options to set the status of the command rule: • DBMS_MACUTL.G_YES or 'y' to enable the command rule (default) • DBMS_MACUTL.G_NO or 'n' to disable the command rule, including the capture of violations in the simulation log • DBMS_MACUTL.G_SIMULATION or 's' to enable SQL statements to execute but capture violations in the simulation log
event_name	An event that the command rule defines. This setting enables the command rule to correspond to an ALTER SYSTEM SET EVENTS *event_name* statement. For example, to track trace events, you would set *event_name* to TRACE.
component_name	A component of the event_name setting. Example settings are DV, OLS, or GCS. You can find valid component names by issuing ORADEBUG DOC COMPONENT RDBMS as user SYS. The output displays parent and child components, which you can use for the component_name setting. For example, both XS (parent) and XSSESSION (child of XS) are valid component names. If you select the parent component, then the command rule applies to it and the child components.
action_name	An action of the component_name setting
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root If you update the common command rule in an application root and want it visible to the associated PDBs, then you must synchronize the application. For example: `ALTER PLUGGABLE DATABASE APPLICATION saas_sales_app SYNC;`
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Example

The following example shows how to update a common system event command rule. This command rule is in the application root, so the user running this procedure must be in the CDB root. Any user names or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE(
 rule_set_name   => 'Disabled',
```

```
 event_name      => 'TRACE',
 component_name  => 'DV',
 enabled         => 'n',
 scope           => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

17

Oracle Database Vault Factor APIs

The DBMS_MACADM PL/SQL package has factor-related Oracle Database Vault rule procedures and functions, and DVF has functions to manage factors.

- DBMS_MACADM Factor Procedures and Functions
  The DBMS_MACADM PL/SQL package provides procedures and functions to configure factors.
- Oracle Database Vault Run-Time PL/SQL Procedures and Functions
  Oracle Database Vault provides procedural interfaces to administer Database Vault security options and manage Database Vault security enforcements.
- Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault maintains the DVF schema functions when you use the DBMS_MACADM PL/SQL package to manage the various factors.

17.1 DBMS_MACADM Factor Procedures and Functions

The DBMS_MACADM PL/SQL package provides procedures and functions to configure factors.

Only users who have been granted the DV_OWNER or DV_ADMIN role can use these procedures and functions.

- ADD_FACTOR_LINK Procedure
  The ADD_FACTOR_LINK procedure specifies a parent-child relationship for two factors.
- ADD_POLICY_FACTOR Procedure
  The ADD_POLICY_FACTOR procedure specifies that the label for a factor contributes to the Oracle Label Security label for a policy.
- CHANGE_IDENTITY_FACTOR Procedure
  The CHANGE_IDENTITY_FACTOR procedure associates an identity with a different factor.
- CHANGE_IDENTITY_VALUE Procedure
  The CHANGE_IDENTITY_FACTOR procedure updates the value of an identity.
- CREATE_DOMAIN_IDENTITY Procedure
  The CREATE_DOMAIN_IDENTITY procedure is used for Oracle Real Application Clusters (Oracle RAC) and Oracle Label Security.
- CREATE_FACTOR Procedure
  The CREATE_FACTOR procedure creates a factor.
- CREATE_FACTOR_TYPE Procedure
  The CREATE_FACTOR_TYPE procedure creates a user-defined factor type.
- CREATE_IDENTITY Procedure
  The CREATE_IDENTITY procedure assigns an identity and an associated trust level for a given factor.
- CREATE_IDENTITY_MAP Procedure
  The CREATE_IDENTITY_MAP procedure defines tests that can derive the identity of a factor from the value of linked child factors (subfactors).

- DELETE_FACTOR Procedure
  The `DELETE_FACTOR` procedure deletes a factor.
- DELETE_FACTOR_LINK Procedure
  The `DELETE_FACTOR_LINK` procedure removes a parent-child relationship for two factors.
- DELETE_FACTOR_TYPE Procedure
  The `DELETE_FACTOR_TYPE` procedure deletes a factor type.
- DELETE_IDENTITY Procedure
  The `DELETE_IDENTITY` procedure removes an identity from an existing factor.
- DELETE_IDENTITY_MAP Procedure
  The `DELETE_IDENTITY_MAP` procedure removes an identity map for a factor.
- DROP_DOMAIN_IDENTITY Procedure
  The `DROP_DOMAIN_IDENTITY` procedure removes an Oracle Real Application Clusters database node from a domain.
- GET_SESSION_INFO Function
  The `GET_SESSION_INFO` function returns information from the `SYS.V_$SESSION` system table for the current session.
- GET_INSTANCE_INFO Function
  The `GET_INSTANCE_INFO` function returns information from the `SYS.V_$INSTANCE` system table about the current database instance.
- RENAME_FACTOR Procedure
  The `RENAME_FACTOR` procedure renames a factor; the name change takes effect everywhere the factor is used.
- RENAME_FACTOR_TYPE Procedure
  The `RENAME_FACTOR` procedure renames a factor type; the name change takes effect everywhere the factor type is used.
- UPDATE_FACTOR Procedure
  The `UPDATE_FACTOR` procedure updates the description of a factor type.
- UPDATE_FACTOR_TYPE Procedure
  The `UPDATE_FACTOR_TYPE` procedure updates a factor type.
- UPDATE_IDENTITY Procedure
  The `UPDATE_IDENTITY` procedure updates the trust level of a factor identity.

Related Topics

- Configuring Factors
  Factors allow you to create and use complex attributes through PL/SQL to make Oracle Database Vault authorization decisions.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

17.1.1 ADD_FACTOR_LINK Procedure

The ADD_FACTOR_LINK procedure specifies a parent-child relationship for two factors.

Syntax

```
DBMS_MACADM.ADD_FACTOR_LINK(
  parent_factor_name IN VARCHAR2,
  child_factor_name  IN VARCHAR2,
  label_indicator    IN VARCHAR2);
```

Parameters

Table 17-1 ADD_FACTOR_LINK Parameters

Parameter	Description
parent_factor_name	Parent factor name. To find existing parent and child factors in the current database instance, query the DBA_DV_FACTOR_LINK view.
child_factor_name	Child factor name.
label_indicator	Indicates that the child factor being linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Specify either DBMS_MACUTL.G_YES (for Yes) or DBMS_MACUTL.G_NO (for No). To find the Oracle Label Security policies and labels associated with factors, query the following views: • DBA_DV_MAC_POLICY: Lists Oracle Label Security policies defined in the current database instance. • DBA_DV_MAC_POLICY_FACTOR: Lists the factors that are associated with Oracle Label Security policies for the current database instance. • DBA_DV_POLICY_LABEL: Lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.

Example

```
BEGIN
 DBMS_MACADM.ADD_FACTOR_LINK(
  parent_factor_name => 'HQ_ClientID',
  child_factor_name  => 'Div1_ClientID',
  label_indicator    => DBMS_MACUTL.G_YES);
END;
/
```

17.1.2 ADD_POLICY_FACTOR Procedure

The ADD_POLICY_FACTOR procedure specifies that the label for a factor contributes to the Oracle Label Security label for a policy.

Syntax

```
DBMS_MACADM.ADD_POLICY_FACTOR(
  policy_name  IN VARCHAR2,
  factor_name  IN VARCHAR2);
```

Parameters

Table 17-2 ADD_POLICY_FACTOR Parameters

Parameter	Description
policy_name	Oracle Label Security policy name. To find the policies defined in the current database instance, query the DBA_DV_MAC_POLICY view. To find factors that are associated with Oracle Label Security policies, query DBA_DV_MAC_POLICY_FACTOR.
factor_name	Factor name. To find existing factors, query the DBA_DV_FACTOR view.

Example

```
BEGIN
 DBMS_MACADM.ADD_POLICY_FACTOR(
  policy_name  => 'AccessData',
  factor_name  => 'Sector2_ClientID');
END;
/
```

17.1.3 CHANGE_IDENTITY_FACTOR Procedure

The CHANGE_IDENTITY_FACTOR procedure associates an identity with a different factor.

Syntax

```
DBMS_MACADM.CHANGE_IDENTITY_FACTOR(
  factor_name     IN VARCHAR2,
  value           IN VARCHAR2,
  new_factor_name IN VARCHAR2);
```

Parameters

Table 17-3 CHANGE_IDENTITY_FACTOR Parameters

Parameter	Description
factor_name	Current factor name. To find existing factors, query the DBA_DV_FACTOR view.

Table 17-3 (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 DBA_DV_IDENTITY view. To find current identity mappings, query the DBA_DV_IDENTITY_MAP view.
new_factor_name	Name of the factor to associate with the identity, which you can find by querying the DBA_DV_FACTOR view.

Example

```
BEGIN
 DBMS_MACADM.CHANGE_IDENTITY_FACTOR(
  factor_name     => 'Sector2_ClientID',
  value           => 'intranet',
  new_factor_name => 'Sector4_ClientID');
END;
/
```

17.1.4 CHANGE_IDENTITY_VALUE Procedure

The CHANGE_IDENTITY_FACTOR procedure updates the value of an identity.

Syntax

```
DBMS_MACADM.CHANGE_IDENTITY_VALUE(
  factor_name  IN VARCHAR2,
  value        IN VARCHAR2,
  new_value    IN VARCHAR2);
```

Parameters

Table 17-4 CHANGE_IDENTITY_VALUE Parameters

Parameter	Description
factor_name	Factor name. To find existing factors, query the DBA_DV_FACTOR view.
value	Current value associated with the identity. To find existing identities for each factor in the current database instance, query the DBA_DV_IDENTITY view. To find current identity mappings, query the DBA_DV_IDENTITY_MAP view.
new_value	New identity value, up to 1024 characters in mixed-case.

Example

```
BEGIN
 DBMS_MACADM.CHANGE_IDENTITY_VALUE(
  factor_name  => 'Sector2_ClientID',
  value        => 'remote',
  new_value    => 'intranet');
```

```
END;
/
```

17.1.5 CREATE_DOMAIN_IDENTITY Procedure

The `CREATE_DOMAIN_IDENTITY` procedure is used for Oracle Real Application Clusters (Oracle RAC) and Oracle Label Security.

It adds an Oracle RAC database node to the domain factor identities and labels it according to an Oracle Label Security policy

Syntax

```
DBMS_MACADM.CREATE_DOMAIN_IDENTITY(
  domain_name  IN VARCHAR2,
  domain_host  IN VARCHAR2,
  policy_name  IN VARCHAR2 DEFAULT NULL,
  domain_label IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 17-5 CREATE_DOMAIN_IDENTITY Parameters

Parameter	Description
`domain_name`	Name of the domain to which to add the host. To find the logical location of the database within the network structure within a distributed database system, run the `DVF.F$DATABASE_DOMAIN` function. See Related Topics.
`domain_host`	Oracle Real Application Clusters host name being added to the domain. To find host name of a database, run the `DVF.F$DATABASE_HOSTNAME` function. See Related Topics.
`policy_name`	Oracle Label Security policy name. If you omit the policy name, then the domain is not associated with any policy. To find the available policies, query the `DBA_DV_MAC_POLICY` view.
`domain_label`	Name of the domain to which to add the Oracle Label Security policy.

Examples

```
BEGIN
 DBMS_MACADM.CREATE_DOMAIN_IDENTITY(
  domain_name  => 'example',
  domain_host  => 'mydom_host',
  policy_name  => 'AccessData',
  domain_label => 'sensitive');
END;
/
```

Related Topics

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

17.1.6 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.

Syntax

```
DBMS_MACADM.CREATE_FACTOR(
  factor_name       IN VARCHAR2,
  factor_type_name  IN VARCHAR2,
  description       IN VARCHAR2,
  rule_set_name     IN VARCHAR2,
  get_expr          IN VARCHAR2,
  validate_expr     IN VARCHAR2,
  identify_by       IN NUMBER,
  labeled_by        IN NUMBER,
  eval_options      IN NUMBER,
  audit_options     IN NUMBER,
  fail_options      IN NUMBER);
```

Parameters

Table 17-6 CREATE_FACTOR Parameters

Parameter	Description
`factor_name`	Factor name, up to 128 characters in mixed-case, without spaces. To find existing factors in the current database instance, query the `DBA_DV_FACTOR` view.
`factor_type_name`	Type of the factor, up to 128 characters in mixed-case, without spaces. To find existing factor types, query the `DBA_DV_FACTOR_TYPE` view, described in .
`description`	Optional description of the purpose of the factor, up to 1024 characters in mixed-case.
`rule_set_name`	Rule set name if you want to use a rule set to control when and how a factor identity is set. To find existing rule sets, query the `DBA_DV_RULE_SET` view. For more details, see the `rule_set_name` description for creating factors. (Refer to Related Topics.)
`get_expr`	Valid PL/SQL expression that retrieves the identity of a factor. It can use up to 255 characters in mixed-case. For more details, see the `get_expr` description for creating factors. (Refer to Related Topics.)
`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. For more details, see the `validate_expr` description for creating factors. (Refer to Related Topics.)

Table 17-6 (Cont.) CREATE_FACTOR Parameters

Parameter	Description
identify_by	Options for determining the identity of a factor, based on the expression set for the get_expr parameter: • DBMS_MACUTL.G_IDENTIFY_BY_CONSTANT: By constant • DBMS_MACUTL.G_IDENTIFY_BY_METHOD: By method • DBMS_MACUTL.G_IDENTIFY_BY_FACTOR: By factor • DBMS_MACUTL.G_IDENTIFY_BY_CONTEXT: By context For more details, see the identify_by description for creating factors. (Refer to Related Topics.)
labeled_by	Options for labeling the factor: • DBMS_MACUTL.G_LABELED_BY_SELF: Labels the identities for the factor directly from the labels associated with an Oracle Label Security policy (default) • DBMS_MACUTL.G_LABELED_BY_FACTORS: Derives the factor identity label from the labels of its child factor identities. For more details, see the labeled_by description for creating factors. (Refer to Related Topics.)
eval_options	Options for evaluating the factor when the user logs on: • DBMS_MACUTL.G_EVAL_ON_SESSION: When the database session is created (default) • DBMS_MACUTL.G_EVAL_ON_ACCESS: Each time the factor is accessed • DBMS_MACUTL.G_EVAL_ON_STARTUP: On start-up For more details, see the eval_options description for creating factors. (Refer to Related Topics.)
audit_options	Omit, or specify DBMS_MACUTL.G_AUDIT_OFF (default). Starting with Oracle Database 23c, traditional auditing is desupported. To audit a factor, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a factor.
fail_options	Options for reporting factor errors: • DBMS_MACUTL.G_FAIL_WITH_MESSAGE: Shows an error message (default) • DBMS_MACUTL.G_FAIL_SILENTLY: Does not show an error message For more details, see the fail_options description for creating factors. (Refer to Related Topics.)

Example

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'Limit_DBA_Access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_METHOD,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_SESSION,
  audit_options     => DBMS_MACUTL.G_AUDIT_OFF,
  fail_options      => DBMS_MACUTL.G_FAIL_SILENTLY);
```

```
END;
/
```

Related Topics

- CREATE_IDENTITY Procedure
  The `CREATE_IDENTITY` procedure assigns an identity and an associated trust level for a given factor.
- 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.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

17.1.7 CREATE_FACTOR_TYPE Procedure

The `CREATE_FACTOR_TYPE` procedure creates a user-defined factor type.

Syntax

```
DBMS_MACADM.CREATE_FACTOR_TYPE(
  name        IN VARCHAR2,
  description IN VARCHAR2);
```

Parameters

Table 17-7 CREATE_FACTOR_TYPE Parameters

Parameter	Description
`name`	Factor type name, up to 128 characters in mixed-case, without spaces. To find existing factor types, query the `DBA_DV_FACTOR_TYPE` view.
`description`	Description of the purpose of the factor type, up to 1024 characters in mixed-case.

Example

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR_TYPE(
  name        => 'Sector2Instance',
  description => 'Checks DB instances used in Sector 2');
END;
/
```

17.1.8 CREATE_IDENTITY Procedure

The `CREATE_IDENTITY` procedure assigns an identity and an associated trust level for a given factor.

After you create a factor, you must assign it an identity.

Syntax

```
DBMS_MACADM.CREATE_IDENTITY(
  factor_name  IN VARCHAR2,
```

```
  value        IN VARCHAR2,
  trust_level  IN NUMBER);
```

Parameters

Table 17-8 CREATE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors, query the DBA_DV_FACTOR view.
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 Related Topics for more information about trust levels and label security.

Example

```
BEGIN
 DBMS_MACADM.CREATE_IDENTITY(
  factor_name  => 'Sector2_ClientID',
  value        => 'intranet',
  trust_level  => 5);
END;
/
```

Related Topics

- Creating and Configuring a Factor Identity
  You can create and configure a factor identity for an existing factor.

17.1.9 CREATE_IDENTITY_MAP Procedure

The CREATE_IDENTITY_MAP procedure defines tests that can derive the identity of a factor from the value of linked child factors (subfactors).

Syntax

```
DBMS_MACADM.CREATE_IDENTITY_MAP(
  identity_factor_name  IN VARCHAR2,
  identity_factor_value IN VARCHAR2,
  parent_factor_name    IN VARCHAR2,
  child_factor_name     IN VARCHAR2,
  operation             IN VARCHAR2,
  operand1              IN VARCHAR2,
  operand2              IN VARCHAR2);
```

Parameters

Table 17-9 CREATE_IDENTITY_MAP Parameters

Parameter	Description
identity_factor_name	Factor the identity map is for. To find existing factors in the current database instance, query the DBA_DV_FACTOR view.
identity_factor_value	Value the factor assumes if the identity map evaluates to TRUE. To find existing factor identities, query the DBA_DV_IDENTITY view. To find current factor identity mappings, use DBA_DV_IDENTITY_MAP.
parent_factor_name	The parent factor link to which the map is related. To find existing parent-child factor mappings, query the DBA_DV_IDENTITY_MAP view.
child_factor_name	The child factor link to which the map is related.
operation	Relational operator for the identity map (for example, <, >, =, and so on).
operand1	Left operand for the relational operator; refers to the low value you enter.
operand2	Right operand for the relational operator; refers to the high value you enter.

Example

```
BEGIN
 DBMS_MACADM.CREATE_IDENTITY_MAP(
  identity_factor_name  => 'Sector2_ClientID',
  identity_factor_value => 'intranet',
  parent_factor_name    => 'HQ_ClientID',
  child_factor_name     => 'Div1_ClientID',
  operation             => '<',
  operand1              => '192.0.2.50',
  operand2              => '192.0.2.100');
END;
/
```

17.1.10 DELETE_FACTOR Procedure

The DELETE_FACTOR procedure deletes a factor.

Syntax

```
DBMS_MACADM.DELETE_FACTOR(
  factor_name IN VARCHAR2);
```

Parameters

Table 17-10 DELETE_FACTOR Parameter

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view.

Example

```
EXEC DBMS_MACADM.DELETE_FACTOR('Sector2_ClientID');
```

17.1.11 DELETE_FACTOR_LINK Procedure

The DELETE_FACTOR_LINK procedure removes a parent-child relationship for two factors.

Syntax

```
DBMS_MACADM.DELETE_FACTOR_LINK(
  parent_factor_name IN VARCHAR2,
  child_factor_name  IN VARCHAR2);
```

Parameters

Table 17-11 DELETE_FACTOR_LINK Parameters

Parameter	Description
parent_factor_name	Factor name. To find factors that are used in parent-child mappings in the current database instance, query the DBA_DV_FACTOR_LINK view.
child_factor_name	Factor name

Example

```
BEGIN
 DBMS_MACADM.DELETE_FACTOR_LINK(
  parent_factor_name => 'HQ_ClientID',
  child_factor_name  => 'Div1_ClientID');
END;
/
```

17.1.12 DELETE_FACTOR_TYPE Procedure

The DELETE_FACTOR_TYPE procedure deletes a factor type.

Syntax

```
DBMS_MACADM.DELETE_FACTOR_TYPE(
  name IN VARCHAR2);
```

Parameters

Table 17-12 DELETE_FACTOR_TYPE Parameters

Parameter	Description
name	Factor type name. To find existing factor types, query the DBA_DV_FACTOR_TYPE view.

Example

```
EXEC DBMS_MACADM.DELETE_FACTOR_TYPE('Sector2Instance');
```

17.1.13 DELETE_IDENTITY Procedure

The DELETE_IDENTITY procedure removes an identity from an existing factor.

Syntax

```
DBMS_MACADM.DELETE_IDENTITY(
  factor_name IN VARCHAR2,
  value       IN VARCHAR2);
```

Parameters

Table 17-13 DELETE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view.
value	Identity value associated with the factor. To find the identities for each factor in the current database instance, query the DBA_DV_IDENTITY view.

Example

```
BEGIN
 DBMS_MACADM.DELETE_IDENTITY(
  factor_name => 'Sector2_ClientID',
  value       => 'intranet');
END;
/
```

17.1.14 DELETE_IDENTITY_MAP Procedure

The DELETE_IDENTITY_MAP procedure removes an identity map for a factor.

Syntax

```
DBMS_MACADM.DELETE_IDENTITY_MAP(
  identity_factor_name  IN VARCHAR2,
  identity_factor_value IN VARCHAR2,
```

```
  parent_factor_name    IN VARCHAR2,
  child_factor_name     IN VARCHAR2,
  operation             IN VARCHAR2,
  operand1              IN VARCHAR2,
  operand2              IN VARCHAR2);
```

Parameters

Table 17-14 DELETE_IDENTITY_MAP Parameters

Parameter	Description
identity_factor_name	Factor the identity map is for. To find existing factors in the current database instance, query the DBA_DV_FACTOR view.
identity_factor_value	Value the factor assumes if the identity map evaluates to TRUE. To find existing factor identities, query the DBA_DV_IDENTITY view. To find current factor identity mappings, query DBA_DV_IDENTITY_MAP.
parent_factor_name	The parent factor link to which the map is related. To find existing parent-child factors, query the DBA_DV_FACTOR view.
child_factor_name	The child factor to which the map is related.
operation	Relational operator for the identity map (for example, <, >, =, and so on).
operand1	Left (low value) operand for the relational operator.
operand2	Right (high value) operand for the relational operator.

Example

```
BEGIN
 DBMS_MACADM.DELETE_IDENTITY_MAP(
  identity_factor_name  => 'Sector2_ClientID',
  identity_factor_value => 'intranet',
  parent_factor_name    => 'HQ_ClientID',
  child_factor_name     => 'Div1_ClientID',
  operation             => '<',
  operand1              => '192.0.2.10',
  operand2              => '192.0.2.15');
END;
/
```

17.1.15 DROP_DOMAIN_IDENTITY Procedure

The DROP_DOMAIN_IDENTITY procedure removes an Oracle Real Application Clusters database node from a domain.

Syntax

```
DBMS_MACADM.DROP_DOMAIN_IDENTITY(
  domain_name   IN VARCHAR2,
  domain_host   IN VARCHAR2);
```

Parameters

Table 17-15 DROP_DOMAIN_IDENTITY Parameters

Parameter	Description
domain_name	Name of the domain to which the host was added. To find the domain of a database as specified by the DB_DOMAIN initialization parameter, run the DVF.F$DATABASE_DOMAIN function.
domain_host	Oracle Real Application Clusters host name being that was added to the domain. To find the host name for a specified database, run the DVF.F$DATABASE_HOSTNAME function.

Example

```
BEGIN
 DBMS_MACADM.DROP_DOMAIN_IDENTITY(
  domain_name  => 'example',
  domain_host  => 'mydom_host');
END;
/
```

Related Topics

- 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.

17.1.16 GET_SESSION_INFO Function

The GET_SESSION_INFO function returns information from the SYS.V_$SESSION system table for the current session.

The V$SESSION data dictionary view also contains session information from this table.

Syntax

```
DBMS_MACADM.GET_SESSION_INFO(
  p_parameter IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 17-16 GET_SESSION_INFO Parameter

Parameter	Description
p_parameter	Column name in the SYS.V_$SESSION system table.

Example

```
DECLARE
 session_var varchar2 := null;
BEGIN
```

```
 session_var = DBMS_MACADM.GET_SESSION_INFO('PROCESS');
END;
/
```

17.1.17 GET_INSTANCE_INFO Function

The `GET_INSTANCE_INFO` function returns information from the `SYS.V_$INSTANCE` system table about the current database instance.

The `V$INSTANCE` data dictionary view also contains database instance information from this table.

Syntax

```
DBMS_MACADM.GET_INSTANCE_INFO(
  p_parameter IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 17-17 GET_INSTANCE_INFO Parameter

Parameter	Description
`p_parameter`	Column name in the `SYS.V_$INSTANCE` system table

Example

```
DECLARE
 instance_var varchar2 := null;
BEGIN
 instance_var = DBMS_MACADM.GET_INSTANCE_INFO('INSTANCE_NAME');
END;
/
```

17.1.18 RENAME_FACTOR Procedure

The `RENAME_FACTOR` procedure renames a factor; the name change takes effect everywhere the factor is used.

Syntax

```
DBMS_MACADM.RENAME_FACTOR(
  factor_name     IN VARCHAR2,
  new_factor_name IN VARCHAR2);
```

Parameters

Table 17-18 RENAME_FACTOR Parameters

Parameter	Description
`factor_name`	Current factor name. To find existing factors in the current database instance, query the `DBA_DV_FACTOR` view.
`new_factor_name`	New factor name, up to 128 characters in mixed-case, without spaces.

Example

```
BEGIN
 DBMS_MACADM.RENAME_FACTOR(
  factor_name     => 'Sector2_ClientID',
  new_factor_name => 'Sector2_Clients');
END;
/
```

17.1.19 RENAME_FACTOR_TYPE Procedure

The `RENAME_FACTOR` procedure renames a factor type; the name change takes effect everywhere the factor type is used.

Syntax

```
DBMS_MACADM.RENAME_FACTOR_TYPE(
  old_name  IN VARCHAR2,
  new_name  IN VARCHAR2);
```

Parameters

Table 17-19 RENAME_FACTOR_TYPE Parameters

Parameter	Description
`old_name`	Current factor type name. To find existing factor types in the current database instance, query the `DBA_DV_FACTOR_TYPE` view.
`new_name`	New factor type name, up to 128 characters in mixed-case, without spaces.

Example

```
BEGIN
 DBMS_MACADM.RENAME_FACTOR_TYPE(
  old_name  => 'Sector2Instance',
  new_name  => 'Sector2DBInstance');
END;
/
```

17.1.20 UPDATE_FACTOR Procedure

The `UPDATE_FACTOR` procedure updates the description of a factor type.

When you update a factor, the existing traditional audit records are disabled. You must create a unified audit policy to capture new audit records.

Syntax

```
DBMS_MACADM.UPDATE_FACTOR(
  factor_name       IN VARCHAR2,
  factor_type_name  IN VARCHAR2,
  description       IN VARCHAR2,
  rule_set_name     IN VARCHAR2,
  get_expr          IN VARCHAR2,
  validate_expr     IN VARCHAR2,
```

```
  identify_by       IN NUMBER,
  labeled_by        IN NUMBER,
  eval_options      IN NUMBER,
  audit_options     IN NUMBER,
  fail_options      IN NUMBER);
```

Parameters

Table 17-20 UPDATE_FACTOR

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view.
factor_type_name	Factor type name. To find existing factor types, query the DBA_DV_FACTOR_TYPE view.
description	Description of the purpose of the factor, up to 1024 characters in mixed-case.
rule_set_name	Name of the rule set used to control when and how a factor identity is set. To find existing rule sets, query the DBA_DV_RULE_SET view .
get_expr	Valid PL/SQL expression that retrieves the identity of a factor. It can use up to 255 characters in mixed-case. 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.
identify_by	Options for determining the identity of a factor, based on the expression set for the get_expr parameter: • DBMS_MACUTL.G_IDENTIFY_BY_CONSTANT: By constant • DBMS_MACUTL.G_IDENTIFY_BY_METHOD: By method • DBMS_MACUTL.G_IDENTIFY_BY_FACTOR: By factor • DBMS_MACUTL.G_IDENTIFY_BY_CONTEXT: By context
labeled_by	Options for labeling the factor: • DBMS_MACUTL.G_LABELED_BY_SELF: Labels the identities for the factor directly from the labels associated with an Oracle Label Security policy • DBMS_MACUTL.G_LABELED_BY_FACTORS: Derives the factor identity label from the labels of its child factor identities. The default for labeled_by is the previously set value, which you can find by querying the DBA_DV_FACTOR data dictionary view.
eval_options	Options for evaluating the factor when the user logs on: • DBMS_MACUTL.G_EVAL_ON_SESSION: When the database session is created • DBMS_MACUTL.G_EVAL_ON_ACCESS: Each time the factor is accessed • DBMS_MACUTL.G_EVAL_ON_STARTUP: On start-up The default for eval_options is the previously set value, which you can find by querying the DBA_DV_FACTOR data dictionary view.

Table 17-20 (Cont.) UPDATE_FACTOR

Parameter	Description
audit_options	Perform one of the following actions: • Set DBMS_MACUTL.G_RULESET_AUDIT_OFF to disable traditional auditing for this factor policy. • Leave the current existing audit_options setting as is if you want to continue using this traditional auditing setting. To do so, you can specify NULL, or specify the same setting as the current existing audit_options for this factor, or simply do not specify audit_options (the default is NULL). Starting with Oracle Database 23c, traditional auditing is desupported. To audit a factor, you must use unified auditing. See *Oracle Database Security Guide* for an example of how to create a unified audit policy for a factor.
fail_options	Options for reporting factor errors: • DBMS_MACUTL.G_FAIL_WITH_MESSAGE: Shows an error message. • DBMS_MACUTL.G_FAIL_SILENTLY: Does not show an error message. The default for fail_options is the previously set value, which you can find by querying the DBA_DV_FACTOR data dictionary view.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'Limit_DBA_Access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_METHOD,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_ACCESS,
  audit_options     => DBMS_MACUTL.G_AUDIT_OFF,
  fail_options      => DBMS_MACUTL.G_FAIL_WITH_MESSAGE);
END;
/
```

If you did not make any modifications to the audit_options parameter of your factor, then the existing traditional audit policy will still be in place. If you updated the audit_options parameter of the factor, then this auditing will be disabled because traditional auditing is desupported starting in release 23c. To capture new audit records, create and enable a unified audit policy. For example:

```
CREATE AUDIT POLICY Sector2_DB_pol
 ACTIONS SELECT, UPDATE, DELETE
 ACTIONS COMPONENT=DV FACTOR ERROR ON "Sector2_DB";

AUDIT POLICY Sector2_DB_pol EXCEPT psmith;
```

You can view the audit records by querying the `UNIFIED_AUDIT_TRAIL` data dictionary view. See How Oracle *Oracle Database Security Guide* for how this works.

Related Topics

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

17.1.21 UPDATE_FACTOR_TYPE Procedure

The `UPDATE_FACTOR_TYPE` procedure updates a factor type.

Syntax

```
DBMS_MACADM.UPDATE_FACTOR_TYPE(
  name          IN VARCHAR2,
  description   IN VARCHAR2);
```

Parameters

Table 17-21 UPDATE_FACTOR_TYPE Parameters

Parameter	Description
`name`	Factor type name. To find existing factor types in the current database instance, query the `DBA_DV_FACTOR_TYPE` view.
`description`	Description of the purpose of the factor type, up to 1024 characters in mixed case.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_FACTOR_TYPE(
  name        => 'Sector2DBInstance',
  description => 'Checks DB instances used in Sector 2');
END;
/
```

17.1.22 UPDATE_IDENTITY Procedure

The `UPDATE_IDENTITY` procedure updates the trust level of a factor identity.

Syntax

```
DBMS_MACADM.UPDATE_IDENTITY(
  factor_name   IN VARCHAR2,
  value         IN VARCHAR2,
  trust_level   IN NUMBER);
```

Parameters

Table 17-22 UPDATE_IDENTITY Parameters

Parameter	Description
factor_name	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view. To find factors that have identities, query DBA_DV_IDENTITY.
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.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_IDENTITY(
  factor_name  => 'Sector2_ClientID',
  value        => 'intranet',
  trust_level  => 10);
END;
/
```

Related Topics

- Creating and Configuring a Factor Identity
  You can create and configure a factor identity for an existing factor.

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

Oracle Database Vault provides procedural interfaces to administer Database Vault security options and manage Database Vault security enforcements.

- About Oracle Database Vault Run-Tine PL/SQL Procedures and Functions
  Oracle Database Vault provides a set of PL/SQL procedures and functions that are specific to factors.
- SET_FACTOR Procedure
  The SET_FACTOR procedure can be exposed to an application that requires the ability to set factor identities dynamically.
- GET_FACTOR Function
  The GET_FACTOR function is exposed to the DVF schema to allow the public factor functions to resolve the identity of a factor. The return type is VARCHAR2.
- GET_FACTOR_LABEL Function
  The GET_FACTOR_LABEL function returns the label for the specified factor when the factor has a label assigned to it for the specified Oracle Label Security policy. The return type is VARCHAR2.

- GET_TRUST_LEVEL Function
  The GET_TRUST_LEVEL function returns the trust level of the current session identity for the factor requested. The return type is VARCHAR2.
- GET_TRUST_LEVEL_FOR_IDENTITY Function
  The GET_TRUST_LEVEL_FOR_IDENTITY function returns the trust level for the factor and identity requested. The return type is VARCHAR2.
- ROLE_IS_ENABLED Function
  The ROLE_IS_ENABLED function returns a boolean value that specifies whether a database role has been enabled. The return type is BOOLEAN.

17.2.1 About Oracle Database Vault Run-Tine PL/SQL Procedures and Functions

Oracle Database Vault provides a set of PL/SQL procedures and functions that are specific to factors.

These procedures and functions that expose the logic to validate a DDL command for realm violations and command authorizations. Additional procedures and functions are provided to set the value of a factor (assuming their associated rule sets evaluate to true) (for example, from a Web application), to retrieve the trust level for a session or specific factor identity, and to get the label for a factor identity. These procedures and functions are provided so that a database administrator does not grant the EXECUTE privilege on all DVSYS package procedures to the general database account population. The procedures and functions expose only the minimum methods that are required. All of these functions and procedures are publicly available for applications that need them.

17.2.2 SET_FACTOR Procedure

The SET_FACTOR procedure can be exposed to an application that requires the ability to set factor identities dynamically.

It wraps the package procedure DBMS_MACADM.SET_FACTOR. When a factor has a rule set associated with it for assignment and if the rule set returns true, then the value is set. Normal rule set handling occurs, and the factor value (identity) validation method is called. This procedure is available (to execute) to the general database account population.

Syntax

```
SET_FACTOR(
  p_factor IN VARCHAR2,
  p_value  IN VARCHAR2);
```

Parameters

Table 17-23 SET_FACTOR Parameters

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR data dictionary view.

Table 17-23 (Cont.) SET_FACTOR Parameters

Parameter	Description
p_value	Identity value, up to 1024 characters in mixed case. To find the identities for each factor in the current database instance, query the DBA_DV_IDENTITY data dictionary view.

Example

```
EXECUTE SET_FACTOR(''Sector2_ClientID'', ''identity'');
```

17.2.3 GET_FACTOR Function

The GET_FACTOR function is exposed to the DVF schema to allow the public factor functions to resolve the identity of a factor. The return type is VARCHAR2.

This function enables the F$ functions in the DVF schema. This function is available (to execute) to the general database account population.

Syntax

```
GET_FACTOR(
  p_factor IN VARCHAR2)
RETURN VARCHAR2;
```

Parameter

Table 17-24 GET_FACTOR Parameter

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR data dictionary view.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Factor Identity',
  rule_expr => 'GET_FACTOR(''Sector2_ClientID'')');
END;
/
```

17.2.4 GET_FACTOR_LABEL Function

The GET_FACTOR_LABEL function returns the label for the specified factor when the factor has a label assigned to it for the specified Oracle Label Security policy. The return type is VARCHAR2.

The function returns a label that is merged with the maximum session label for the policy if the policy is configured with Oracle Label Security. The function is available (to execute) to the general database population.

Syntax

```
GET_FACTOR_LABEL(
  p_factor      IN VARCHAR2,
  p_policy_name IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 17-25 GET_FACTOR_LABEL Parameters

Parameter	Description
p_factor	Factor name. To find the available factors in the current database instance, query the DBA_DV_FACTOR data dictionary view. To find factors that are associated with Oracle Label Security policies, use DBA_DV_MAC_POLICY_FACTOR.
p_policy_name	Oracle Label Security policy name. Use the following data dictionary views to find information about policies and factors in the current database instance: • DBA_DV_MAC_POLICY: Lists Oracle Label Security policies defined in the current database instance. • DBA_DV_MAC_POLICY_FACTOR: Lists the factors that are associated with Oracle Label Security policies for the current database instance. • DBA_DV_POLICY_LABEL: Lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get the ClientID Factor Label',
  rule_expr => 'GET_FACTOR_LABEL(''Sector2_ClientID'', ''Access Locations'')');
END;
/
```

17.2.5 GET_TRUST_LEVEL Function

The GET_TRUST_LEVEL function returns the trust level of the current session identity for the factor requested. The return type is VARCHAR2.

This function is available (to execute) to the general database account population.

Syntax

```
GET_TRUST_LEVEL(
  p_factor IN VARCHAR2)
RETURN VARCHAR2;
```

Parameter

Table 17-26 GET_TRUST_LEVEL Parameter

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR data dictionary view.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Trust Level',
  rule_expr => 'GET_TRUST_LEVEL(''Sector2_ClientID'')');
END;
/
```

Related Topics

- Creating and Configuring a Factor Identity
  You can create and configure a factor identity for an existing factor.

17.2.6 GET_TRUST_LEVEL_FOR_IDENTITY Function

The GET_TRUST_LEVEL_FOR_IDENTITY function returns the trust level for the factor and identity requested. The return type is VARCHAR2.

This function is available (to execute) to the general database account population.

Syntax

```
GET_TRUST_LEVEL_FOR_IDENTITY(
  p_factor   IN VARCHAR2,
  p_identity IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 17-27 GET_TRUST_LEVEL_FOR_IDENTITY Parameters

Parameter	Description
p_factor	Factor name. To find existing factors in the current database instance, query the DBA_DV_FACTOR view.
p_identity	Identity value. To find the identities for each factor in the current database instance, use the DBA_DV_IDENTITY data dictionary view.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Client ID Identity Trust Level',
```

```
  rule_expr => 'GET_TRUST_LEVEL_FOR_IDENTITY(''Sector2_ClientID'',
''identity'')');
END;
/
```

17.2.7 ROLE_IS_ENABLED Function

The `ROLE_IS_ENABLED` function returns a boolean value that specifies whether a database role has been enabled. The return type is `BOOLEAN`.

This function is available (to execute) to the general database account population.

Syntax

```
ROLE_IS_ENABLED(
  p_role IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 17-28 ROLE_IS_ENABLED Parameter

Parameter	Description
`p_role`	Database role name to check. To find existing roles, use the following data dictionary views: • `DBA_ROLES`: Finds available roles in the current database instance. • `DBA_DV_REALM_AUTH`: Finds the authorization of a particular role. • `DBA_DV_ROLE`: Finds existing secure application roles used in privilege management.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check if SYSADM Role Is Enabled',
  rule_expr => 'ROLE_IS_ENABLED(''SYSADM'')');
END;
/
```

17.3 Oracle Database Vault DVF PL/SQL Factor Functions

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

- About Oracle Database Vault DVF PL/SQL Factor Functions
  Oracle Database Vault provides DVF factor-specific functions for frequently used activities.
- F$AUTHENTICATION_METHOD Function
  The `F$AUTHENTICATION_METHOD` function returns the method of authentication in `VARCHAR2` data type.
- F$CLIENT_IP Function
  The `F$CLIENT_IP` function returns the IP address of the computer from which the client is connected, in `VARCHAR2` data type.

- F$DATABASE_DOMAIN Function
  The `F$DATABASE_DOMAIN` function returns the domain of the database as specified in the `DB_DOMAIN` initialization parameter, in `VARCHAR2` data type.
- F$DATABASE_HOSTNAME Function
  The `F$DATABASE_HOSTNAME` function returns the host name of the computer on which the instance is running, in `VARCHAR2` data type.
- F$DATABASE_INSTANCE Function
  The `F$DATABASE_INSTANCE` function returns the instance identification number of the current database instance, in `VARCHAR2` data type.
- F$DATABASE_IP Function
  The `F$DATABASE_IP` function returns the IP address of the computer on which the database instance is running, in `VARCHAR2` data type.
- F$DATABASE_NAME Function
  The `F$DATABASE_NAME` function returns the name of the database as specified in the `DB_NAME` initialization parameter, in `VARCHAR2` data type.
- F$DOMAIN Function
  The `F$DOMAIN` function returns a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level. The return type is `VARCHAR2`.
- FDV_CLIENT_IDENTIFIER Function
  The `F$DV$_CLIENT_IDENTIFIER` function returns an Oracle Database Vault client identifier.
- FDV_DBLINK_INFO Function
  The `F$DV$_DBLINK_INFO` function returns information about an Oracle Database Vault database link.
- FDV_MODULE Function
  The `F$DV$_MODULE` function returns information about an Oracle Database Vault module.
- F$ENTERPRISE_IDENTITY Function
  The `F$ENTERPRISE_IDENTITY` function returns the enterprise-wide identity for a user, in `VARCHAR2` data type.
- F$IDENTIFICATION_TYPE Function
  The `F$IDENTIFICATION_TYPE` function returns the way the schema of a user was created in the database. Specifically, it reflects the `IDENTIFIED` clause in the `CREATE/ALTER USER` syntax. The return type is `VARCHAR2`.
- F$LANG Function
  The `F$LANG` function returns the ISO abbreviation for the language name, a shorter form than the existing `LANGUAGE` parameter, for the session of the user. The return type is `VARCHAR2`.
- F$LANGUAGE Function
  The `F$LANGUAGE` function returns the language and territory currently used by a user session, along with the database character set. The return type is `VARCHAR2`.
- F$MACHINE Function
  The `F$MACHINE` function returns the computer (host) name for the database client that established the database session. The return type is `VARCHAR2`.

- F$NETWORK_PROTOCOL Function
  The `F$NETWORK_PROTOCOL` function returns the network protocol being used for communication, as specified in the `PROTOCOL=protocol` portion of the connect string. The return type is `VARCHAR2`.
- F$PROXY_ENTERPRISE_IDENTITY Function
  The `F$PROXY_ENTERPRISE_IDENTITY` function returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user. The return type is `VARCHAR2`.
- F$PROXY_USER Function
  The `F$PROXY_USER` function returns the name of a proxy user.
- F$SESSION_USER Function
  The `F$SESSION_USER` function returns the database user name by which the current user is authenticated. This value remains the same throughout the session. The return type is `VARCHAR2`.

17.3.1 About Oracle Database Vault DVF PL/SQL Factor Functions

Oracle Database Vault provides DVF factor-specific functions for frequently used activities.

In addition to the functions and procedures made available from the `DVSYS` schema, the `DVF` schema contains a single function for each factor defined in the system.

The functions are then available to the general database account population through PL/SQL functions and standard SQL. This enables factors to be used in Oracle Label Security, Oracle Virtual Private Database (VPD), and so on.

Typically, you can incorporate these functions into rule expressions. For example:

The functions are then available to the general database account population through PL/SQL functions and standard SQL. This enables factors to be used in Oracle Label Security, Oracle Virtual Private Database (VPD), and so on.

Typically, you can incorporate these functions into rule expressions. For example:

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
 rule_name => 'Not Internal DBA',
 rule_expr => 'DVF.F$SESSION_USER NOT IN (''JSMTIH'', ''TBROWN'')');
END;
/
```

To find the value of a factor function, select from the `DUAL` system table. For example:

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

F$SESSION_USER
------------------------------------------------
SEC_ADMIN_OWEN
```

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;
```

17.3.2 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`
- Transport Layer Security (TLS)-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 TLS.

Syntax

```
DVF.F$AUTHENTICATION_METHOD ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check TLS Authentication Method',
  rule_expr => 'DVF.F$AUTHENTICATION_METHOD = ''SSL''');
END;
/
```

17.3.3 F$CLIENT_IP Function

The `F$CLIENT_IP` function returns the IP address of the computer from which the client is connected, in `VARCHAR2` data type.

Syntax

```
DVF.F$CLIENT_IP ()
RETURN VARCHAR2;
```

Parameters

None

Example

The following example shows how to use `DVF.F$CLIENT_IP` in a rule creation statement. Note that you can only enter one IP address, not a range of IP addresses.

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client IP Address',
  rule_expr => 'DVF.F$CLIENT_IP = ''192.0.2.10''');
END;
/
```

17.3.4 F$DATABASE_DOMAIN Function

The `F$DATABASE_DOMAIN` function returns the domain of the database as specified in the `DB_DOMAIN` initialization parameter, in `VARCHAR2` data type.

Syntax

```
DVF.F$DATABASE_DOMAIN ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client Database Domain',
  rule_expr => 'DVF.F$DATABASE_DOMAIN NOT IN (''EXAMPLE'', ''YOURDOMAIN'')');
END;
/
```

17.3.5 F$DATABASE_HOSTNAME Function

The `F$DATABASE_HOSTNAME` function returns the host name of the computer on which the instance is running, in `VARCHAR2` data type.

Syntax

```
DVF.F$DATABASE_HOSTNAME ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Host Name',
  rule_expr => 'DVF.F$DATABASE_HOSTNAME IN (''SHOBEEN'', ''MAU'')');
END;
/
```

17.3.6 F$DATABASE_INSTANCE Function

The `F$DATABASE_INSTANCE` function returns the instance identification number of the current database instance, in `VARCHAR2` data type.

Syntax

```
DVF.F$DATABASE_INSTANCE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database Instance ID',
  rule_expr => 'DVF.F$DATABASE_INSTANCE = ''SALES_DB''');
END;
/
```

17.3.7 F$DATABASE_IP Function

The `F$DATABASE_IP` function returns the IP address of the computer on which the database instance is running, in `VARCHAR2` data type.

Syntax

```
DVF.F$DATABASE_IP ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database IP address',
  rule_expr => 'DVF.F$DATABASE_IP = ''192.0.2.5''');
END;
/
```

17.3.8 F$DATABASE_NAME Function

The `F$DATABASE_NAME` function returns the name of the database as specified in the `DB_NAME` initialization parameter, in `VARCHAR2` data type.

Syntax

```
DVF.F$DATABASE_NAME ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database DB_NAME Name',
  rule_expr => 'DVF.F$DATABASE_NAME = ''ORCL''');
END;
/
```

17.3.9 F$DOMAIN Function

The `F$DOMAIN` function returns a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level. The return type is `VARCHAR2`.

You can identify a domain using factors such as host name, IP address, and database instance names of the Oracle Database Vault nodes in a secure access path to the database. Each domain can be uniquely determined using a combination of the factor identifiers that identify the domain. You can use these identifying factors and possibly additional factors to define the Maximum Security Label within the domain. This restricts data access and commands, depending on the physical factors about the Oracle Database Vault session. Example domains of interest may be Corporate Sensitive, Internal Public, Partners, and Customers.

Syntax

```
DVF.F$DOMAIN ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Domain',
  rule_expr => 'DVF.F$DOMAIN = ''EXAMPLE.COM''');
END;
/
```

17.3.10 FDV_CLIENT_IDENTIFIER Function

The `F$DV$_CLIENT_IDENTIFIER` function returns an Oracle Database Vault client identifier.

Syntax

```
DVF.F$DV$_CLIENT_IDENTIFIER ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check Database Vault Client Identifiers',
   rule_expr => 'DVF.F$DV$_CLIENT_IDENTIFIER = ''14903BUA765454''';
END;/
```

17.3.11 FDV_DBLINK_INFO Function

The `F$DV$_DBLINK_INFO` function returns information about an Oracle Database Vault database link.

Syntax

```
DVF.F$DV$_DBLINK_INFO ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check Database Vault database link info',
   rule_expr => 'DVF.F$DV$_DBLINK_INFO = ''SOURCE_GLOBAL_NAME=SALES.US.EXAMPLE.COM,
DBLINK_NAME=PDB2_LINK, SOURCE_AUDIT_SESSIONID=200057''';
END;/
```

17.3.12 FDV_MODULE Function

The `F$DV$_MODULE` function returns information about an Oracle Database Vault module.

Syntax

```
DVF.F$DV$_MODULE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check Database Vault modules',
   rule_expr => 'DVF.F$DV$_MODULE = ''SQL*Plus''';
END;/
```

17.3.13 F$ENTERPRISE_IDENTITY Function

The `F$ENTERPRISE_IDENTITY` function returns the enterprise-wide identity for a user, in `VARCHAR2` data type.

- For enterprise users: the Oracle Internet Directory DN.
- For external users: the external identity (Kerberos principal name, Radius and DCE schema names, operating system user name, certificate DN).
- For local users and `SYSDBA`/`SYSOPER` logins: NULL.

The value of the attribute differs by proxy method:

- For a proxy with DN: the Oracle Internet Directory DN of the client.
- For a proxy with certificate: the certificate DN of the client for external users; the Oracle Internet Directory DN for global users.
- For a proxy with user name: the Oracle Internet Directory DN if the client is an enterprise user; NULL if the client is a local database user.

Syntax

```
DVF.F$ENTERPRISE_IDENTITY ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check User Enterprise Identity',
  rule_expr => 'DVF.F$ENTERPRISE_IDENTITY NOT IN (''JSMITH'', ''TSMITH'')');
END;
/
```

17.3.14 F$IDENTIFICATION_TYPE Function

The `F$IDENTIFICATION_TYPE` function returns the way the schema of a user was created in the database. Specifically, it reflects the `IDENTIFIED` clause in the `CREATE`/`ALTER USER` syntax. The return type is `VARCHAR2`.

In the list that follows, the syntax used during schema creation is followed by the identification type returned:

- `IDENTIFIED BY` *`password`*`: LOCAL`
- `IDENTIFIED EXTERNALLY: EXTERNAL`
- `IDENTIFIED GLOBALLY: GLOBAL SHARED`
- `IDENTIFIED GLOBALLY AS DN: GLOBAL PRIVATE`

Syntax

```
DVF.F$IDENTIFICATION_TYPE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check User Schema Creation Type',
  rule_expr => 'DVF.F$IDENTIFICATION_TYPE = ''GLOBAL SHARED''');
END;
/
```

17.3.15 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`.

Syntax

```
DVF.F$LANG ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check ISO Abbreviated Language Name',
  rule_expr => 'DVF.F$LANG IN (''EN'', ''DE'', ''FR'')');
END;
/
```

Related Topics

- *Oracle Database Globalization Support Guide*

17.3.16 F$LANGUAGE Function

The `F$LANGUAGE` function returns the language and territory currently used by a user session, along with the database character set. The return type is `VARCHAR2`.

The return type is in the following format:

language_territory.characterset

Syntax

```
DVF.F$LANGUAGE ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Session Language and Territory',
  rule_expr => 'DVF.F$LANGUAGE = ''AMERICAN_AMERICA.WE8ISO8859P1''');
END;
/
```

Related Topics

- *Oracle Database Globalization Support Guide*

17.3.17 F$MACHINE Function

The `F$MACHINE` function returns the computer (host) name for the database client that established the database session. The return type is `VARCHAR2`.

Syntax

```
DVF.F$MACHINE ()
RETURN VARCHAR2;
```

Parameter

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Client Computer Host Name',
  rule_expr => 'DVF.F$MACHINE NOT IN (''SHOBEEN'', ''SEBASTIAN'')');
END;
/
```

17.3.18 F$NETWORK_PROTOCOL Function

The `F$NETWORK_PROTOCOL` function returns the network protocol being used for communication, as specified in the `PROTOCOL=`*`protocol`* portion of the connect string. The return type is `VARCHAR2`.

Syntax

```
DVF.F$NETWORK_PROTOCOL ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Network Protocol',
  rule_expr => 'DVF.F$NETWORK_PROTOCOL = ''TCP''');
END;
/
```

17.3.19 F$PROXY_ENTERPRISE_IDENTITY Function

The `F$PROXY_ENTERPRISE_IDENTITY` function returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user. The return type is `VARCHAR2`.

Syntax

```
DVF.F$PROXY_ENTERPRISE_IDENTITY ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get OID DN of Enterprise User',
  rule_expr => 'DVF.F$PROXY_ENTERPRISE_IDENTITY = ''cn=Provisioning Admins''');
END;
/
```

17.3.20 F$PROXY_USER Function

The `F$PROXY_USER` function returns the name of a proxy user.

Syntax

```
DVF.PROXY_USER ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
  DBMS_MACADM.CREATE_RULE(
   rule_name => 'Check Proxy Users',
   rule_expr => 'DVF.PROXY_USER NOT IN (''ECHICHESTER'', ''PFITCH'')');
END;/
```

17.3.21 F$SESSION_USER Function

The `F$SESSION_USER` function returns the database user name by which the current user is authenticated. This value remains the same throughout the session. The return type is `VARCHAR2`.

Syntax

```
DVF.F$SESSION_USER ()
RETURN VARCHAR2;
```

Parameters

None

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Check Database User Name',
  rule_expr => 'DVF.F$SESSION_USER IN (''JSMITH'', ''TSMITH'')');
END;
/
```

18
Oracle Database Vault Secure Application Role APIs

The `DBMS_MACADM` and `DBMS_MACSEC_ROLES` PL/SQL packages manage Database Vault secure application roles.

- DBMS_MACADM Secure Application Role Procedures
  The `DBMS_MACADM` package creates, renames, assigns, unassigns, updates, and deletes Oracle Database Vault secure application roles.
- DBMS_MACSEC_ROLES Secure Application Role Procedure and Function
  The `DBMS_MACSEC_ROLES` package checks the authorization for users and sets Oracle Database Vault secure application roles.

Related Topics

- Configuring Secure Application Roles for Oracle Database Vault
  Secure application roles enable you to control how much access users have to an application.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

18.1 DBMS_MACADM Secure Application Role Procedures

The `DBMS_MACADM` package creates, renames, assigns, unassigns, updates, and deletes Oracle Database Vault secure application roles.

- CREATE_ROLE Procedure
  The `CREATE_ROLE` procedure creates an Oracle Database Vault secure application role.
- DELETE_ROLE Procedure
  The `DELETE_ROLE` procedure deletes an Oracle Database Vault secure application role.
- RENAME_ROLE Procedure
  The `RENAME_ROLE` procedure renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.
- UPDATE_ROLE Procedure
  The `UPDATE_ROLE` procedure updates a Oracle Database Vault secure application role.

18.1.1 CREATE_ROLE Procedure

The `CREATE_ROLE` procedure creates an Oracle Database Vault secure application role.

Syntax

```
DBMS_MACADM.CREATE_ROLE(
  role_name      IN VARCHAR2,
  enabled        IN VARCHAR2,
  rule_set_name  IN VARCHAR2);
```

Parameters

Table 18-1 CREATE_ROLE Parameters

Parameter	Description
role_name	Role name, up to 128 characters, with no spaces. Prepend the role name with c## or C## if it is a common role. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view.
enabled	DBMS_MACUTL.G_YES makes the role available for enabling; DBMS_MACUTL.G_NO prevents the role from being enabled. The default is DBMS_MACUTL.G_YES.
rule_set_name	Name of rule set to determine whether this secure application can be enabled. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view.

Example

```
BEGIN
 DBMS_MACADM.CREATE_ROLE(
  role_name      => 'Sector2_APP_MGR',
  enabled        => DBMS_MACUTL.G_YES,
  rule_set_name  => 'Check App2 Access');
END;
/
```

18.1.2 DELETE_ROLE Procedure

The DELETE_ROLE procedure deletes an Oracle Database Vault secure application role.

Syntax

```
DBMS_MACADM.DELETE_ROLE(
  role_name IN VARCHAR2);
```

Parameters

Table 18-2 DELETE_ROLE Parameter

Parameter	Description
role_name	Role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view.

Example

```
EXEC DBMS_MACADM.DELETE_ROLE('SECT2_APP_MGR');
```

18.1.3 RENAME_ROLE Procedure

The RENAME_ROLE procedure renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.

Syntax

```
DBMS_MACADM.RENAME_ROLE(
  role_name      IN VARCHAR2,
  new_role_name  IN VARCHAR2);
```

Parameters

Table 18-3 RENAME_ROLE Parameters

Parameter	Description
role_name	Current role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view.
new_role_name	Role name, up to 128 characters, with no spaces. Ensure that this name follows the standard Oracle naming conventions for role creation described in *Oracle Database SQL Language Reference*. Prepend the role name with c## or C## if it is a common role.

Example

```
BEGIN
 DBMS_MACADM.RENAME_ROLE(
  role_name      => 'SECT2_APP_MGR',
  new_role_name  => 'SECT2_SYSADMIN');
END;
/
```

18.1.4 UPDATE_ROLE Procedure

The UPDATE_ROLE procedure updates a Oracle Database Vault secure application role.

Syntax

```
DBMS_MACADM.UPDATE_ROLE(
  role_name      IN VARCHAR2,
  enabled        IN VARCHAR2,
  rule_set_name  IN VARCHAR2);
```

Parameters

Table 18-4 UPDATE_ROLE Parameters

Parameter	Description
role_name	Role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view.

Table 18-4 (Cont.) UPDATE_ROLE Parameters

Parameter	Description
enabled	DBMS_MACUTL.G_YES (Yes) makes the role available for enabling; DBMS_MACUTL.G_NO (No) prevents the role from being enabled. The default for enabled is the previously set value, which you can find by querying the DBA_DV_ROLE data dictionary view.
rule_set_name	Name of rule set to determine whether this secure application can be enabled. To find existing rule sets in the current database instance, query the DBA_DV_RULE_SET view.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_ROLE(
  role_name      => 'SECT2_SYSADMIN',
  enabled        => DBMS_MACUTL.G_YES,
  rule_set_name  => 'System Access Controls');
END;
/
```

18.2 DBMS_MACSEC_ROLES Secure Application Role Procedure and Function

The DBMS_MACSEC_ROLES package checks the authorization for users and sets Oracle Database Vault secure application roles.

The DBMS_MACSEC_ROLES package is available to all users.

- CAN_SET_ROLE Function
  The CAN_SET_ROLE function checks if the user invoking the method is authorized to use an Oracle Database Vault secure application role.
- SET_ROLE Procedure
  The SET_ROLE procedure issues the SET ROLE PL/SQL statement for specified roles.

18.2.1 CAN_SET_ROLE Function

The CAN_SET_ROLE function checks if the user invoking the method is authorized to use an Oracle Database Vault secure application role.

The authorization is determined by checking the rule set associated with the role. The return type is BOOLEAN.

Syntax

```
DBMS_MACSEC_ROLES.CAN_SET_ROLE(
  p_role IN VARCHAR2)
RETURN BOOLEAN;
```

Parameters

Table 18-5 CAN_SET_ROLE Parameter

Parameter	Description
p_role	Role name. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACSEC_ROLES.CAN_SET_ROLE('SECTOR2_APP_MGR')
  THEN DBMS_OUTPUT.PUT_LINE('''SECTOR2_APP_MGR'' can be enabled.');
 END IF;
END;
/
```

18.2.2 SET_ROLE Procedure

The SET_ROLE procedure issues the SET ROLE PL/SQL statement for specified roles.

This procedure includes both Oracle Database Vault secure application roles and regular Oracle Database roles in its checking process.

This procedure sets an Oracle Database Vault secure application role only if the rule set that is associated with the role evaluates to true. Before SET ROLE is issued, the CAN_SET_ROLE method is called to check the rule set associated with the role. Run-time rule set behavior such as auditing, failure processing, and event handling occur during this process.

The SET_ROLE procedure is available to the general database account population.

Syntax

```
DBMS_MACSEC_ROLES.SET_ROLE(
  p_role IN VARCHAR2);
```

Parameters

Table 18-6 SET_ROLE Parameter

Parameter	Description
p_role	Role names. You can enter multiple roles, separated by commas (,), including secure application roles and regular roles. To find existing secure application roles in the current database instance, query the DBA_DV_ROLE view. To find all of the existing roles in the database, query the DBA_ROLES data dictionary view.

Example

```
EXEC DBMS_MACSEC_ROLES.SET_ROLE('SECTOR2_APP_MGR, APPS_MGR');
```

You can enter the name of the role in any case (for example, `Sector2_APP_MGR`).

19

Oracle Database Vault Oracle Label Security APIs

You can use the `DBMS_MACADM` PL/SQL package to manage Oracle Label Security labels and policies in Oracle Database Vault.

- CREATE_MAC_POLICY Procedure
  The `CREATE_MAC_POLICY` procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.
- CREATE_POLICY_LABEL Procedure
  The `CREATE_POLICY_LABEL` procedure labels an identity within an Oracle Label Security policy.
- DELETE_MAC_POLICY_CASCADE Procedure
  The `DELETE_MAC_POLICY_CASCADE` procedure deletes all Oracle Database Vault objects related to an Oracle Label Security policy.
- DELETE_POLICY_FACTOR Procedure
  The `DELETE_POLICY_FACTOR` procedure removes the factor from contributing to the Oracle Label Security label.
- DELETE_POLICY_LABEL Procedure
  The `DELETE_POLICY_LABEL` procedure removes the label from an identity within an Oracle Label Security policy.
- UPDATE_MAC_POLICY Procedure
  The `UPDATE_MAC_POLICY` procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Related Topics

- Integrating Oracle Database Vault with Other Oracle Products
  You can integrate Oracle Database Vault with other Oracle products, such as Oracle Data Guard.
- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

19.1 CREATE_MAC_POLICY Procedure

The `CREATE_MAC_POLICY` procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Syntax

```
DBMS_MACADM.CREATE_MAC_POLICY(
  policy_name  IN VARCHAR2,
  algorithm    IN VARCHAR2);
```

Parameters

Table 19-1 CREATE_MAC_POLICY Parameters

Parameter	Description
`policy_name`	Name of an existing policy. To find existing policies in the current database instance, query the `DBA_DV_MAC_POLICY` view.
`algorithm`	Merge algorithm for cases when Oracle Label Security has merged two labels. Enter the code listed in Table 19-2 that corresponds to the merge algorithm you want. For example, enter `HUU` to if you want to select the Maximum Level/Union/Union merge algorithm.

Table 19-2 Oracle Label Security Merge Algorithm Codes

Code	Value
`HUU`	Maximum Level/Union/Union
`HIU`	Maximum Level/Intersection/Union
`HMU`	Maximum Level/Minus/Union
`HNU`	Maximum Level/Null/Union
`HUI`	Maximum Level/Union/Intersection
`HII`	Maximum Level/Intersection/Intersection
`HMI`	Maximum Level/Minus/Intersection
`HNI`	Maximum Level/Null/Intersection
`HUM`	Maximum Level/Union/Minus
`HIM`	Maximum Level/Intersection/Minus
`HMM`	Maximum Level/Minus/Minus
`HNM`	Maximum Level/Null/Minus
`HUN`	Maximum Level/Union/Null
`HIN`	Maximum Level/Intersection/Null
`HMN`	Maximum Level/Minus/Null
`HNN`	Maximum Level/Null/Null
`LUU`	Minimum Level/Union/Union
`LIU`	Minimum Level/Intersection/Union
`LMU`	Minimum Level/Minus/Union
`LNU`	Minimum Level/Null/Union
`LUI`	Minimum Level/Union/Intersection
`LII`	Minimum Level/Intersection/Intersection
`LMI`	Minimum Level/Minus/Intersection
`LNI`	Minimum Level/Null/Intersection
`LUM`	Minimum Level/Union/Minus

Table 19-2 (Cont.) Oracle Label Security Merge Algorithm Codes

Code	Value
LIM	Minimum Level/Intersection/Minus
LMM	Minimum Level/Minus/Minus
LNM	Minimum Level/Null/Minus
LUN	Minimum Level/Union/Null
LIN	Minimum Level/Intersection/Null
LMN	Minimum Level/Minus/Null
LNN	Minimum Level/Null/Null

Example

```
BEGIN
 DBMS_MACADM.CREATE_MAC_POLICY(
  policy_name  => 'Access Locations',
  algorithm    => 'HUU');
END;
/
```

19.2 CREATE_POLICY_LABEL Procedure

The `CREATE_POLICY_LABEL` procedure labels an identity within an Oracle Label Security policy.

Syntax

```
DBMS_MACADM.CREATE_POLICY_LABEL(
  identity_factor_name   IN VARCHAR2,
  identity_factor_value  IN VARCHAR2,
  policy_name            IN VARCHAR2,
  label                  IN VARCHAR2);
```

Parameters

Table 19-3 CREATE_POLICY_LABEL Parameters

Parameter	Description
identity_factor_name	Name of the factor being labeled. To find existing factors in the current database instance, query the `DBA_DV_FACTOR` view. To find factors that are associated with Oracle Label Security policies, use `DBA_DV_MAC_POLICY_FACTOR`.
identity_factor_value	Value of identity for the factor being labeled. To find the identities of existing factors in the current database instance, query the `DBA_DV_IDENTITY` view.
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the `DBA_DV_MAC_POLICY` view.

Table 19-3 (Cont.) CREATE_POLICY_LABEL Parameters

Parameter	Description
label	Oracle Label Security label name. To find existing policy labels for factor identifiers, query the DBA_DV_POLICY_LABEL view.

Example

```
BEGIN
 DBMS_MACADM.CREATE_POLICY_LABEL(
  identity_factor_name   => 'App_Host_Name',
  identity_factor_value  => 'Sect2_Fin_Apps',
  policy_name            => 'Access Locations',
  label                  => 'Sensitive');
END;
/
```

19.3 DELETE_MAC_POLICY_CASCADE Procedure

The DELETE_MAC_POLICY_CASCADE procedure deletes all Oracle Database Vault objects related to an Oracle Label Security policy.

Syntax

```
DBMS_MACADM.DELETE_MAC_POLICY_CASCADE(
  policy_name  IN VARCHAR2);
```

Parameters

Table 19-4 DELETE_MAC_POLICY_CASCADE Parameter

Parameter	Description
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DBA_DV_MAC_POLICY view.

Example

```
EXEC DBMS_MACADM.DELETE_MAC_POLICY_CASCADE('Access Locations');
```

19.4 DELETE_POLICY_FACTOR Procedure

The DELETE_POLICY_FACTOR procedure removes the factor from contributing to the Oracle Label Security label.

Syntax

```
DBMS_MACADM.DELETE_POLICY_FACTOR(
  policy_name  IN VARCHAR2,
  factor_name  IN VARCHAR2);
```

Parameters

Table 19-5 DELETE_POLICY_FACTOR Parameters

Parameter	Description
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DBA_DV_MAC_POLICY view.
factor_name	Name of factor associated with the Oracle Label Security label. To find factors that are associated with Oracle Label Security policies, query DBA_DV_MAC_POLICY_FACTOR.

Example

```
BEGIN
 DBMS_MACADM.DELETE_POLICY_FACTOR(
  policy_name  => 'Access Locations',
  factor_name  => 'App_Host_Name');
END;
/
```

19.5 DELETE_POLICY_LABEL Procedure

The DELETE_POLICY_LABEL procedure removes the label from an identity within an Oracle Label Security policy.

Syntax

```
DBMS_MACADM.DELETE_POLICY_LABEL(
  identity_factor_name   IN VARCHAR2,
  identity_factor_value  IN VARCHAR2,
  policy_name            IN VARCHAR2,
  label                  IN VARCHAR2);
```

Parameters

Table 19-6 DELETE_POLICY_LABEL Parameters

Parameter	Description
identity_factor_name	Name of the factor that was labeled. To find existing factors in the current database instance that are associated with Oracle Label Security policies, query DBA_DV_MAC_POLICY_FACTOR.
identity_factor_value	Value of identity for the factor that was labeled. To find the identities of existing factors in the current database instance, query the DBA_DV_IDENTITY view.
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DBA_DV_MAC_POLICY view.

Table 19-6 (Cont.) DELETE_POLICY_LABEL Parameters

Parameter	Description
label	Oracle Label Security label name. To find existing policy labels for factor identifiers, query the DBA_DV_POLICY_LABEL view.

Example

```
BEGIN
 DBMS_MACADM.DELETE_POLICY_LABEL(
  identity_factor_name   => 'App_Host_Name',
  identity_factor_value  => 'Sect2_Fin_Apps',
  policy_name            => 'Access Locations',
  label                  => 'Sensitive');
END;
/
```

19.6 UPDATE_MAC_POLICY Procedure

The UPDATE_MAC_POLICY procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

Syntax

```
DBMS_MACADM.UPDATE_MAC_POLICY(
  policy_name  IN VARCHAR2,
  algorithm    IN VARCHAR2);
```

Parameters

Table 19-7 UPDATE_MAC_POLICY

Parameter	Description
policy_name	Name of an existing policy. To find existing policies in the current database instance, query the DBA_DV_MAC_POLICY view.
algorithm	Merge algorithm for cases when Oracle Label Security has merged two labels. See the codes listed in the DBMS_MACADM.CREATE_MAC_POLICY description.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_MAC_POLICY(
  policy_name  => 'Access Locations',
  algorithm    => 'LUI');
END;
/
```

Related Topics

- CREATE_MAC_POLICY Procedure
  The `CREATE_MAC_POLICY` procedure specifies the algorithm to merge labels when computing the label for a factor, or the Oracle Label Security Session label.

20

Oracle Database Vault Utility APIs

Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

- DBMS_MACUTL Constants
  You can use a set of constants, available in the `DBMS_MACUTL` PL/SQL package.
- DBMS_MACUTL Package Procedures and Functions
  The `DBMS_MACUTL` PL/SQL package can perform tasks such as finding a time value or whether a user has the the appropriate privileges.

20.1 DBMS_MACUTL Constants

You can use a set of constants, available in the `DBMS_MACUTL` PL/SQL package.

- DBMS_MACUTL Listing of Constants
  The `DBMS_MACUTL` PL/SQL package provides constants (fields) to use with Oracle Database Vault PL/SQL packages.
- Example: Creating a Realm Using DBMS_MACUTL Constants
  Constants can be used to answer simple Yes or No settings when you create objects in Oracle Database Vault.
- Example: Creating a Rule Set Using DBMS_MACUTL Constants
  Constants can be used to set options such as handler options.
- Example: Creating a Factor Using DBMS_MACUTL Constants
  Constants can be used to set information specific to factors, such as identity or labeling.

20.1.1 DBMS_MACUTL Listing of Constants

The `DBMS_MACUTL` PL/SQL package provides constants (fields) to use with Oracle Database Vault PL/SQL packages.

Table 20-1 summarizes constant (that is, fields) descriptions for the `DBMS_MACUTL` package.

Many of these constants have equivalents in the Oracle Database Vault package. For example, the `enabled` parameter, which is available in several procedures, can accept either `Y` (for Yes) or the constant `G_YES`. Choosing one over the other is a matter of personal preference. They both have the same result.

Table 20-1 DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
`G_ALL_OBJECT`	`VARCHAR2(1)`	Used with the realm API `object_name` and `object_type` parameters as a wildcard to indicate all object names or all object types.

Table 20-1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_AUDIT_ALWAYS	NUMBER	Used with the factor API audit_options parameter to enable traditional auditing. Starting with Oracle Database 23c, traditional auditing is desupported; you cannot set this constant.
G_AUDIT_OFF	NUMBER	Used with the factor API audit_options parameter to disable traditional auditing. Because traditional auditing is desupported, this is the only audit constant that you can set.
G_AUDIT_ON_GET_ERROR	NUMBER	Used with the factor API audit_options parameter to audit using traditional auditing if the expression specified in the get_expr parameter returns an error. Because traditional auditing is desupported, you cannot set this constant.
G_AUDIT_ON_GET_NULL	NUMBER	Used with the factor API audit_options parameter to audit using traditional auditing if the expression in the get_expr field is null. Because traditional auditing is desupported, you cannot set this constant.
G_AUDIT_ON_TRUST_LEVEL_NEG	NUMBER	Used with the factor API audit_options parameter to audit using traditional auditing if the trust level is negative. Because traditional auditing is desupported, you cannot set this constant.
G_AUDIT_ON_TRUST_LEVEL_NULL	NUMBER	Used with the factor API audit_options parameter to audit using traditional auditing if no trust level exists. Because traditional auditing is desupported, you cannot set this constant.
G_AUDIT_ON_VALIDATE_ERROR	NUMBER	Used with the factor API audit_options parameter to audit using traditional auditing if the validation function returns an error. Starting with Oracle Database 23c, traditional auditing is desupported; you cannot set this constant.

Table 20-1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
G_AUDIT_ON_VALIDATE_FALSE	NUMBER	Used with the factor API audit_options parameter to audit using traditional auditing if validation function is false. Starting with Oracle Database 23c, traditional auditing is desupported; you cannot set this constant.
G_DISABLE	NUMBER	Used to disable Oracle Database Vault policies and command rules
G_ENABLE	NUMBER	Used to enable Oracle Database Vault policies and command rules
G_EVAL_ON_ACCESS	NUMBER	Used with the factor API eval_options parameter to reevaluate the factor each time it is accessed.
G_EVAL_ON_SESSION	NUMBER	Used with the factor API eval_options parameter to evaluate the factor only once, when the user logs in to the session.
G_FAIL_SILENTLY	NUMBER	Used with the fail_options parameter to fail and show no error message.
G_FAIL_WITH_MESSAGE	NUMBER	Used with the fail_options parameter to fail and show an error message.
G_IDENTIFY_BY_CONSTANT	NUMBER	Used with the factor API identify_by parameter: Fixed value in PL/SQL expression defined in the get_expr parameter.
G_IDENTIFY_BY_CONTEXT	NUMBER	Used with the factor API identify_by parameter to indicate context.
G_IDENTIFY_BY_FACTOR	NUMBER	Used with the factor API identify_by parameter for subfactors through the factor_link$ table.
G_IDENTIFY_BY_METHOD	NUMBER	Used with the factor API identify_by parameter: Expression in get_expr field
G_IDENTIFY_BY_RULESET	NUMBER	Used with the factor API identify_by parameter: Expression and Rule Set with the factor_expr$ table
G_LABELED_BY_FACTORS	NUMBER	Used with the factor API labeled_by parameter to derive the label from subfactor and merge algorithm.
G_LABELED_BY_SELF	NUMBER	Used with the factor API labeled_by parameter to label the factor identities.
G_MAX_SESSION_LABEL	VARCHAR2(30)	This is the highest label a user could set based on the factors. It does not consider the label for a user.

Table 20-1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
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_PARTIAL	NUMBER	Sets the enforcement state of the realms and command rules under an Oracle Database Vault policy to be changed individually
G_REALM_AUDIT_OFF	NUMBER	Used with the realm API audit_options parameter to disable auditing. Because traditional auditing is desupported, this is the only realm audit option that you can set.
G_REALM_AUDIT_SUCCESS	NUMBER	Used with the realm API audit_options parameter: Audit on successful realm access Because traditional auditing is desupported, you cannot set this constant.
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_OFF	NUMBER	Used with the rule set API audit_options parameter to disable auditing. Because traditional auditing is desupported, this is the only rule set constant that you can set.
G_RULESET_EVAL_ALL	NUMBER	Used with the rule set API eval_options parameter to enable the rule set to succeed if all rules evaluate to true.
G_RULESET_EVAL_ANY	NUMBER	Used with the rule set API eval_options parameter to succeed if any of the rules evaluate to true.

Table 20-1 (Cont.) DBMS_MACUTL Listing of Constants

Constant Name	Data Type	Description
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_SIMULATION	NUMBER	Used to set the enforcement state of a policy to simulation mode. This mode does not raise errors for realm or command rule violations. Instead, an error is logged in a designated log table with sufficient information relevant to the error (for example, users or SQL command.)
G_USER_POLICY_LABEL	VARCHAR2(30)	This is what Oracle Label Security has decided the user's label should be set to after factoring in the preceding values.
G_YES	VARCHAR2(1)	Used with the following APIs: • The factor API label_indicator parameter to indicate that a child factor linked to a parent factor contributes to the label of the parent factor in an Oracle Label Security integration. • Any API that uses the enabled parameter.

Related Topics

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

20.1.2 Example: Creating a Realm Using DBMS_MACUTL Constants

Constants can be used to answer simple Yes or No settings when you create objects in Oracle Database Vault.

Example 20-1 shows how to use the `G_YES DBMS_MACUTL` constant when creating a realm.

Example 20-1 Creating a Realm Using DBMS_MACUTL Constants

```
BEGIN
 DBMS_MACADM.CREATE_REALM(
  realm_name    => 'Performance Statistics Realm',
  description   => 'Realm to measure performance',
  enabled       =>  DBMS_MACUTL.G_YES,
  audit_options =>  DBMS_MACUTL.G_REALM_AUDIT_OFF);
END;
/
```

20.1.3 Example: Creating a Rule Set Using DBMS_MACUTL Constants

Constants can be used to set options such as handler options.

Example 20-2 shows how to use several `DBMS_MACUTL` constants when creating a rule set.

Example 20-2 Creating a Rule Set Using DBMS_MACUTL Constants

```
BEGIN
 DBMS_MACADM.CREATE_RULE_SET(
  rule_set_name    => 'Limit_DBA_Access',
  description      => 'DBA access through predefined processes',
  enabled          => DBMS_MACUTL.G_YES,
  eval_options     => DBMS_MACUTL.G_RULESET_EVAL_ALL,
  audit_options    => DBMS_MACUTL.G_RULESET_AUDIT_OFF,
  fail_options     => DBMS_MACUTL.G_RULESET_FAIL_SHOW,
  fail_message     => 'Rule Set Limit_DBA_Access has failed.',
  fail_code        => 20000,
  handler_options  => DBMS_MACUTL.G_RULESET_HANDLER_FAIL,
  handler          => 'dbavowner.email_alert');
END;
/
```

20.1.4 Example: Creating a Factor Using DBMS_MACUTL Constants

Constants can be used to set information specific to factors, such as identity or labeling.

Example 20-3 shows how to use constants when creating a factor.

Example 20-3 Creating a Factor Using DBMS_MACUTL Constants

```
BEGIN
 DBMS_MACADM.CREATE_FACTOR(
  factor_name       => 'Sector2_DB',
  factor_type_name  => 'Instance',
  description       => ' ',
  rule_set_name     => 'DB_access',
  get_expr          => 'UPPER(SYS_CONTEXT(''USERENV'',''DB_NAME''))',
  validate_expr     => 'dbavowner.check_db_access',
  identify_by       => DBMS_MACUTL.G_IDENTIFY_BY_FACTOR,
  labeled_by        => DBMS_MACUTL.G_LABELED_BY_SELF,
  eval_options      => DBMS_MACUTL.G_EVAL_ON_SESSION,
```

```
  audit_options     => DBMS_MACUTL.G_AUDIT_OFF,
  fail_options      => DBMS_MACUTL.G_FAIL_SILENTLY);
END;
/
```

20.2 DBMS_MACUTL Package Procedures and Functions

The DBMS_MACUTL PL/SQL package can perform tasks such as finding a time value or whether a user has the the appropriate privileges.

- CHECK_DVSYS_DML_ALLOWED Procedure
  The CHECK_DVSYS_DML_ALLOWED procedure checks if a user can issue Data Modification Language (DML) commands to access the DVSYS objects.
- GET_CODE_VALUE Function
  The GET_CODE_VALUE function finds the value for a code within a code group, and then returns a VARCHAR2 value.
- GET_SECOND Function
  The GET_SECOND function returns the seconds in Oracle SS (seconds) format (00–59), and then returns a NUMBER value.
- GET_MINUTE Function
  The GET_MINUTE function returns the minute in Oracle MI (minute) format (00–59), in a NUMBER value.
- GET_HOUR Function
  The GET_HOUR function returns the hour in Oracle HH24 (hour) format (00–23), in a NUMBER value.
- GET_DAY Function
  The GET_DAY function returns the day in Oracle DD (day) format (01–31), in a NUMBER value.
- GET_MONTH Function
  The GET_MONTH function returns the month in Oracle MM (month) format (01–12), in a NUMBER value.
- GET_YEAR Function
  The GET_YEAR function returns the year in Oracle YYYY (year) format (0001–9999), in a NUMBER value.
- IS_ALPHA Function
  The IS_ALPHA function returns a BOOLEAN value indicating if a character is alphabetic.
- IS_DIGIT Function
  The IS_DIGIT function checks returns a BOOLEAN value indicating if a character is numeric.
- IS_DVSYS_OWNER Function
  The IS_DVSYS_OWNER function returns a BOOLEAN value indicating if a user is authorized to manage the Oracle Database Vault configuration.
- IS_OLS_INSTALLED Function
  The IS_OLS_INSTALLED function returns a BOOLEAN value indicating if Oracle Label Security is installed.
- IS_OLS_INSTALLED_VARCHAR Function
  The IS_OLS_INSTALLED_VARCHAR function returns a BOOLEAN value indicating if Oracle Label Security is installed.

- ROLE_GRANTED_ENABLED_VARCHAR Function
  The ROLE_GRANTED_ENABLED_VARCHAR function returns a VARCHAR2 value indicating the role grant and enablement status of a user.
- USER_HAS_OBJECT_PRIVILEGE Function
  The USER_HAS_OBJECT_PRIVILEGE function returns a BOOLEAN value indicating if user or role can access an object through a single specified object privilege grant.
- USER_HAS_ROLE Function
  The USER_HAS_ROLE function returns a BOOLEAN value indicating if a user has a role privilege, directly or indirectly (through another role).
- USER_HAS_ROLE_VARCHAR Function
  The USER_HAS_ROLE_VARCHAR function returns a VARCHAR2 value indicating if a user has a role privilege, directly or indirectly (through another role).
- USER_HAS_SYSTEM_PRIVILEGE Function
  The USER_HAS_SYSTEM_PRIVILEGE function returns a BOOLEAN value indicating if a user has a system privilege, directly or indirectly (through a role).

20.2.1 CHECK_DVSYS_DML_ALLOWED Procedure

The CHECK_DVSYS_DML_ALLOWED procedure checks if a user can issue Data Modification Language (DML) commands to access the DVSYS objects.

Syntax

```
DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED(
  p_user IN VARCHAR2 DEFAULT USER);
```

Parameter

Table 20-2 CHECK_DVSYS_DML_ALLOWED Parameter

Parameter	Description
p_user	User to check. To find existing users in the current database instance, query the following views: • DBA_USERS: Finds available users for the current database instance. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management.

Example

User SYSTEM fails the check:

```
EXEC DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED('system');

ERROR at line 1:
ORA-47920: Authorization failed for user system to perform this operation
ORA-06512: at "DBMS_MACUTL", line 23
ORA-06512: at "DBMS_MACUTL", line 372
ORA-06512: at "DBMS_MACUTL", line 508
ORA-06512: at "DBMS_MACUTL", line 572
ORA-06512: at line 1
```

User sec_admin_owen, who has the DV_OWNER role, passes the check:

```
EXEC DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED('ec_admin_owen');

PL/SQL procedure successfully completed.
```

20.2.2 GET_CODE_VALUE Function

The GET_CODE_VALUE function finds the value for a code within a code group, and then returns a VARCHAR2 value.

Syntax

```
DBMS_MACUTL.GET_CODE_VALUE(
  p_code_group IN VARCHAR2,
  p_code       IN VARCHAR2)
RETURN VARCHAR2;
```

Parameters

Table 20-3 GET_CODE_VALUE Parameters

Parameter	Description
p_code_group	Code group (for example, AUDIT_EVENTS or BOOLEAN). To find available code groups in the current database instance, query the DBA_DV_CODE view.
p_code	ID of the code. This ID is listed when you run the DBA_DV_CODE view.

Example

```
BEGIN
 DBMS_MACADM.CREATE_RULE(
  rule_name => 'Get Label Algorithm for Maximum Level/Union/Null',
  rule_expr => 'DBMS_MACUTL.GET_CODE_VALUE(''LABEL_ALG'', ''HUN'') = ''Union''');
END;
/
```

20.2.3 GET_SECOND Function

The GET_SECOND function returns the seconds in Oracle SS (seconds) format (00–59), and then returns a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_SECOND(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 20-4 GET_SECOND Parameter

Parameter	Description
p_date	Date in SS format (for example, 59). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the date and time of the PDB that uses the time zone of the PDB.

Example

```
SET SERVEROUTPUT ON
DECLARE
  seconds number;
BEGIN
  seconds := DBMS_MACUTL.GET_SECOND(TO_DATE('03-APR-2009 6:56 PM',
  'dd-mon-yyyy hh:mi PM'));
  DBMS_OUTPUT.PUT_LINE('Seconds: '||seconds);
END;
/
```

This example, which uses a fixed date and time, returns the following:

```
Seconds: 56
```

20.2.4 GET_MINUTE Function

The GET_MINUTE function returns the minute in Oracle MI (minute) format (00–59), in a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_MINUTE(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 20-5 GET_MINUTE Parameter

Parameter	Description
p_date	Date in MI format (for example, 30, as in 2:30). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the date and time of the PDB that uses the time zone of the PDB.

Example

```
SET SERVEROUTPUT ON
DECLARE
  minute number;
```

```
BEGIN
  minute := DBMS_MACUTL.GET_MINUTE(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Minute: '||minute);
END;
/
```

Output similar to the following appears:

```
Minute: 17
```

20.2.5 GET_HOUR Function

The `GET_HOUR` function returns the hour in Oracle HH24 (hour) format (00–23), in a `NUMBER` value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_HOUR(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 20-6 GET_HOUR Parameter

Parameter	Description
`p_date`	Date in HH24 format (for example, `14` for 2:00 p.m.) If you do not specify a date, then Oracle Database Vault uses the Oracle Database `SYSDATE` function to retrieve the date and time of the PDB that uses the time zone of the PDB.

Example

```
SET SERVEROUTPUT ON
DECLARE
  hours number;
BEGIN
  hours := DBMS_MACUTL.GET_HOUR(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Hour: '||hours);
END;
/
```

Output similar to the following appears:

```
Hour: 12
```

20.2.6 GET_DAY Function

The `GET_DAY` function returns the day in Oracle DD (day) format (01–31), in a `NUMBER` value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_DAY(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 20-7 GET_DAY Parameter

Parameter	Description
p_date	Date in DD format (for example, 01 for the first day of the month). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the date and time of the PDB that uses the time zone of the PDB.

Example

```
SET SERVEROUTPUT ON
DECLARE
  day number;
BEGIN
  day := DBMS_MACUTL.GET_DAY(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Day: '||day);
END;
/
```

Output similar to the following appears:

```
Day: 3
```

20.2.7 GET_MONTH Function

The GET_MONTH function returns the month in Oracle MM (month) format (01–12), in a NUMBER value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_MONTH(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 20-8 GET_MONTH Parameter

Parameter	Description
p_date	Date in MM format (for example, 08 for the month of August). If you do not specify a date, then Oracle Database Vault uses the Oracle Database SYSDATE function to retrieve the date and time of the PDB that uses the time zone of the PDB.

Example

```
SET SERVEROUTPUT ON
DECLARE
  month number;
BEGIN
  month := DBMS_MACUTL.GET_MONTH(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Month: '||month);
END;
/
```

Output similar to the following appears:

```
Month: 4
```

20.2.8 GET_YEAR Function

The `GET_YEAR` function returns the year in Oracle YYYY (year) format (0001–9999), in a `NUMBER` value.

It is useful for rule expressions based on time data.

Syntax

```
DBMS_MACUTL.GET_YEAR(
  p_date IN DATE DEFAULT SYSDATE)
RETURN NUMBER;
```

Parameter

Table 20-9 GET_YEAR Parameter

Parameter	Description
`p_date`	Date in YYYY format (for example, `1984`). If you do not specify a date, then Oracle Database Vault uses the `SYSDATE` function to retrieve the date and time of the PDB that uses the time zone of the PDB.

Example

```
SET SERVEROUTPUT ON
DECLARE
  year number;
BEGIN
  year := DBMS_MACUTL.GET_YEAR(SYSDATE);
  DBMS_OUTPUT.PUT_LINE('Year: '||year);
END;
/
```

20.2.9 IS_ALPHA Function

The `IS_ALPHA` function returns a `BOOLEAN` value indicating if a character is alphabetic.

`IS_ALPHA` returns `TRUE` if the character is alphabetic.

Syntax

```
DBMS_MACUTL.IS_ALPHA(
  c IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 20-10 IS_ALPHA Parameter

Parameter	Description
c	String with one character

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_ALPHA('z')
  THEN DBMS_OUTPUT.PUT_LINE('The alphabetic character was found');
 ELSE
  DBMS_OUTPUT.PUT_LINE('No alphabetic characters today.');
 END IF;
END;
/
```

20.2.10 IS_DIGIT Function

The `IS_DIGIT` function checks returns a `BOOLEAN` value indicating if a character is numeric.

`IS_DIGIT` returns `TRUE` if the character is a digit.

Syntax

```
DBMS_MACUTL.IS_DIGIT(
  c IN VARCHAR2)
RETURN BOOLEAN;
```

Parameter

Table 20-11 IS_DIGIT Parameter

Parameter	Description
c	String with one character

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_DIGIT('7')
  THEN DBMS_OUTPUT.PUT_LINE('The numeric character was found');
 ELSE
  DBMS_OUTPUT.PUT_LINE('No numeric characters today.');
 END IF;
```

```
END;
/
```

20.2.11 IS_DVSYS_OWNER Function

The `IS_DVSYS_OWNER` function returns a `BOOLEAN` value indicating if a user is authorized to manage the Oracle Database Vault configuration.

`IS_DVSYS_OWNER` returns `TRUE` if the user is authorized.

Syntax

```
DBMS_MACUTL.IS_DVSYS_OWNER(
  p_user IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameter

Table 20-12 IS_DVSYS_OWNER Parameter

Parameter	Description
`p_user`	User to check. To find existing users, query the following data dictionary views: • `DBA_USERS`: Finds available users for the current database instance. • `DBA_DV_REALM_AUTH`: Finds the authorization of a particular user or role. • `DBA_DV_ROLE`: Finds existing secure application roles used in privilege management.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_DVSYS_OWNER('PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('PSMITH is authorized to manage Database Vault.');
 ELSE
  DBMS_OUTPUT.PUT_LINE('PSMITH is not authorized to manage Database Vault.');
 END IF;
END;
/
```

20.2.12 IS_OLS_INSTALLED Function

The `IS_OLS_INSTALLED` function returns a `BOOLEAN` value indicating if Oracle Label Security is installed.

If Oracle Label Security is installed, `IS_OLS_INSTALLED` returns `TRUE`.

Syntax

```
DBMS_MACUTL.IS_OLS_INSTALLED()
RETURN BOOLEAN;
```

Parameters

None

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_OLS_INSTALLED()
  THEN DBMS_OUTPUT.PUT_LINE('OLS is installed');
 ELSE
  DBMS_OUTPUT.PUT_LINE('OLS is not installed');
 END IF;
END;
/
```

20.2.13 IS_OLS_INSTALLED_VARCHAR Function

The `IS_OLS_INSTALLED_VARCHAR` function returns a `BOOLEAN` value indicating if Oracle Label Security is installed.

If Oracle Label Security is installed, then `IS_OLS_INSTALLED_VARCHAR` returns `Y`.

Syntax

```
DBMS_MACUTL.IS_OLS_INSTALLED_VARCHAR()
RETURN VARCHAR2;
```

Parameters

None

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.IS_OLS_INSTALLED()
  THEN DBMS_OUTPUT.PUT_LINE('OLS is installed');
 ELSE
  DBMS_OUTPUT.PUT_LINE('OLS is not installed');
 END IF;
END;
/
```

20.2.14 ROLE_GRANTED_ENABLED_VARCHAR Function

The `ROLE_GRANTED_ENABLED_VARCHAR` function returns a `VARCHAR2` value indicating the role grant and enablement status of a user.

`ROLE_GRANTED_ENABLED_VARCHAR` function checks whether a user has a role granted directly or indirectly (through another role) with a sufficient scope or the role currently is enabled in the session while the role is not granted. If either of these conditions are true, then it returns `Y`.

Because the `SYS_SESSION_ROLES` namespace of the `SYS_CONTEXT` function does not represent the logged in user roles when it is evaluated as a `DVSYS` command rule, Oracle recommends that you use the `ROLE_GRANTED_ENABLED_VARCHAR` function to check if a role is enabled for a logged in user.

Syntax

```
DBMS_MACUTL.ROLE_GRANTED_ENABLED_VARCHAR(
  p_role IN VARCHAR2,
  p_user IN VARCHAR2 DEFAULT USER,
  p_profile IN NUMBER(38) DEFAULT 1,
  p_scope IN VARCHAR2 DEFAULT LOCAL)
RETURN VARCHAR2;
```

Parameters

Table 20-13 ROLE_GRANTED_ENABLED_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. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management.
p_user	User to check. If you want to use ROLE_GRANTED_ENABLED_VARCHAR function as part of a rule evaluation, then you cannot set p_user to CURRENT_USER when ROLE_GRANTED_ENABLED_VARCHAR is being evaluated as an Oracle Database Vault rule. Instead, you can use the SYS_CONTEXT function USERENV namespace SESSION_USER to represent the login user. To find existing users, query the following views: • DBA_USERS: Finds available users for the current database instance. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role.
p_profile	If you are using privilege analysis and the role being checked is used, then specify 1 so that privilege analysis can capture the usage of the role. Otherwise, enter 0.
p_scope	Specify either COMMON for a commonly granted role, or LOCAL for a locally granted role.

Example

This example shows how to use the DBMS_MACUTL.ROLE_GRANTED_ENABLED_VARCHAR function in a command rule to check if the logged in user has the enabled role of EMPLOYEE.

```
BEGIN
DBMS_MACADM.CREATE_RULE(
  rule_name  => 'does role exist',
  rule_expr  => 'DVSYS.DBMS_MACUTL.ROLE_GRANTED_ENABLED_VARCHAR(''EMPLOYEE'',''"''||
dvsys.dv_login_user||''"'') = ''Y''');
END;
/
```

20.2.15 USER_HAS_OBJECT_PRIVILEGE Function

The USER_HAS_OBJECT_PRIVILEGE function returns a BOOLEAN value indicating if user or role can access an object through a single specified object privilege grant.

If the user or role has the object privilege, then USER_HAS_OBJECT_PRIVILEGE returns TRUE.

Syntax

```
DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE(
  p_user          VARCHAR2,
  p_object_owner  VARCHAR2,
  p_object_name   VARCHAR2,
  p_privilege     VARCHAR2)
RETURNS BOOLEAN;
```

Parameters

Table 20-14 USER_HAS_OBJECT_PRIVILEGE Parameters

Parameter	Description
p_user	User or role to check. To find existing users, query they following views: • DBA_USERS: Finds available users for the current database instance. • DBA_ROLES: Finds available roles in the current database instance. • DVA_DV_REALM_AUTH: Finds the authorization of a particular user or role. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management.
p_object_owner	Object owner, such as a schema. To find the available users, query they DBA_USERS view. To find the authorization of a particular user, query they DVA_DV_REALM_AUTH view.
p_object_name	Object name, such as a table within the schema specified in the p_object_owner parameter. To find the available objects, query the ALL_OBJECTS view. To find objects that are secured by existing realms, query they DBA_DV_REALM_OBJECT view.
p_privilege	Object privilege, such as, UPDATE. To find privileges for a database account excluding PUBLIC privileges, query they DBA_DV_USER_PRIVS view. To find all privileges for a database account, query the DBA_DV_USER_PRIVS_ALL. view

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE(
   'SECTOR2_APP_MGR', 'OE', 'ORDERS', 'UPDATE')
  THEN DBMS_OUTPUT.PUT_LINE('SECTOR2_APP_MGR has the UPDATE privilege for the
OE.ORDERS table');
   ELSE
  DBMS_OUTPUT.PUT_LINE('SECTOR2_APP_MGR does not have the UPDATE privilege for
the OE.ORDERS table.');
 END IF;
END;
/
```

20.2.16 USER_HAS_ROLE Function

The USER_HAS_ROLE function returns a BOOLEAN value indicating if a user has a role privilege, directly or indirectly (through another role).

If the user has a role privilege, then USER_HAS_ROLE returns TRUE.

Syntax

```
DBMS_MACUTL.USER_HAS_ROLE(
  p_role IN VARCHAR2,
  p_user IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameters

Table 20-15 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. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management.
p_user	User to check. To find existing users, query the following views: • DBA_USERS: Finds available users for the current database instance. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.USER_HAS_ROLE('SECTOR2_APP_MGR', 'PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('User PSMITH has the SECTOR2_APP_MGR role');
   ELSE
  DBMS_OUTPUT.PUT_LINE('User PSMITH does not have the SECTOR2_APP_MGR role.');
 END IF;
END;
/
```

20.2.17 USER_HAS_ROLE_VARCHAR Function

The USER_HAS_ROLE_VARCHAR function returns a VARCHAR2 value indicating if a user has a role privilege, directly or indirectly (through another role).

If the user has the role privilege specified, then USER_HAS_ROLE_VARCHAR returns Y.

Syntax

```
DBMS_MACUTL.USER_HAS_ROLE_VARCHAR(
  p_role IN VARCHAR2,
```

```
  p_user IN VARCHAR2 DEFAULT USER)
RETURN VARCHAR2;
```

Parameters

Table 20-16 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. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role. • DBA_DV_ROLE: Finds existing secure application roles used in privilege management.
p_user	User to check. To find existing users, query the following views: • DBA_USERS: Finds available users for the current database instance. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role.

20.2.18 USER_HAS_SYSTEM_PRIVILEGE Function

The USER_HAS_SYSTEM_PRIVILEGE function returns a BOOLEAN value indicating if a user has a system privilege, directly or indirectly (through a role).

If the user has the system privilege specified, then USER_HAS_SYSTEM_PRIVILEGE returns TRUE.

Syntax

```
DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE(
  p_privilege  IN VARCHAR2,
  p_user       IN VARCHAR2 DEFAULT USER)
RETURN BOOLEAN;
```

Parameters

Table 20-17 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 DBA_DV_USER_PRIVS view. To find all privileges for a database account, use DBA_DV_USER_PRIVS_ALL.
p_user	User to check. To find existing users, query the following views: • DBA_USERS: Finds available users for the current database instance. • DBA_DV_REALM_AUTH: Finds the authorization of a particular user or role.

Example

```
SET SERVEROUTPUT ON
BEGIN
 IF DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE('EXECUTE', 'PSMITH')
  THEN DBMS_OUTPUT.PUT_LINE('User PSMITH has the EXECUTE ANY PRIVILEGE privilege.');
   ELSE
  DBMS_OUTPUT.PUT_LINE('User PSMITH does not have the EXECUTE ANY PRIVILEGE
privilege.');
 END IF;
END;
/
```

21
Oracle Database Vault General Administrative APIs

The DBMS_MACADM PL/SQL package and the CONFIGURE_DV standalone procedure enable you to you perform general maintenance tasks.

- DBMS_MACADM General System Maintenance Procedures
  The DBMS_MACADM PL/SQL package general system maintenance procedures perform tasks such as authorizing users or adding new language to Oracle Database Vault.
- CONFIGURE_DV General System Maintenance Procedure
  The CONFIGURE_DV procedure configures the initial two Oracle Database user accounts, which are granted the DV_OWNER and DV_ACCTMGR roles, respectively.

21.1 DBMS_MACADM General System Maintenance Procedures

The DBMS_MACADM PL/SQL package general system maintenance procedures perform tasks such as authorizing users or adding new language to Oracle Database Vault.

- ADD_APP_EXCEPTION Procedure
  The ADD_APP_EXCEPTION procedure enables a common user or package to access local schemas.
- ADD_NLS_DATA Procedure
  The ADD_NLS_DATA procedure adds a new language to Oracle Database Vault.
- ALLOW_COMMON_OPERATION Procedure
  The ALLOW_COMMON_OPERATION procedure controls the access that a local user has on common objects in a PDB.
- AUTH_DATAPUMP_CREATE_USER Procedure
  The AUTH_DATAPUMP_CREATE_USER procedure authorizes an Oracle Data Pump user to create users during an Oracle Data Pump import operation.
- AUTH_DATAPUMP_GRANT Procedure
  The AUTH_DATAPUMP_GRANT procedure authorizes an Oracle Data Pump user to grant Oracle Database Vault-protected roles and system privileges during an Oracle Data Pump import operation.
- AUTH_DATAPUMP_GRANT_ROLE Procedure
  The AUTH_DATAPUMP_GRANT_ROLE procedure authorizes an Oracle Data Pump user to grant a specific role during an Oracle Data Pump import operation.
- AUTH_DATAPUMP_GRANT_SYSPRIV Procedure
  The AUTH_DATAPUMP_GRANT_SYSPRIV procedure authorizes an Oracle Data Pump user to grant system privileges during an Oracle Data Pump import operation.

- AUTHORIZE_AUDIT_ADMIN Procedure
  The AUTHORIZE_AUDIT_ADMIN procedure enables a user to perform the auditing tasks that are available to users who have been granted the Oracle Database AUDIT_ADMIN role, such as creating audit policies.
- AUTHORIZE_AUDIT_VIEWER Procedure
  The AUTHORIZE_AUDIT_VIEWER procedure enables a user to perform the auditing tasks that are available to users who have been granted the AUDIT_VIEWER role to query audit-related views.
- AUTHORIZE_DATAPUMP_USER Procedure
  The AUTHORIZE_DATAPUMP_USER procedure authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled.
- AUTHORIZE_DBCAPTURE Procedure
  The AUTHORIZE_DBCAPTURE procedure grants a user authorization to perform Oracle Database Replay workload capture operations.
- AUTHORIZE_DBREPLAY Procedure
  The AUTHORIZE_DBREPLAY procedure grants a user authorization to perform Oracle Database Replay workload replay operations.
- AUTHORIZE_DDL Procedure
  The AUTHORIZE_DDL procedure grants a user authorization to run Data Definition Language (DDL) statements on the specified schema.
- AUTHORIZE_DIAGNOSTIC_ADMIN Procedure
  The AUTHORIZE_DIAGNOSTIC_ADMIN procedure authorizes a user to query diagnostic views and tables.
- AUTHORIZE_MAINTENANCE_USER Procedure
  The AUTHORIZE_MAINTENANCE_USER procedure grants a user authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.
- AUTHORIZE_PREPROCESSOR Procedure
  The AUTHORIZE_PREPROCESSOR procedure grants a user authorization to run preprocessor programs through external tables.
- AUTHORIZE_PROXY_USER Procedure
  The AUTHORIZE_PROXY_USER procedure grants a proxy user authorization to proxy other user accounts, as long as the proxy user has database authorization.
- AUTHORIZE_SCHEDULER_USER Procedure
  The AUTHORIZE_SCHEDULER_USER procedure grants a user authorization to schedule database jobs when Oracle Database Vault is enabled.
- AUTHORIZE_SQL_FIREWALL Procedure
  The AUTHORIZE_SQL_FIREWALL procedure grants a user authorization to use SQL Firewall.
- AUTHORIZE_TTS_USER Procedure
  The AUTHORIZE_TTS_USER procedure authorizes a user to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled.
- DELETE_APP_EXCEPTION Procedure
  The DELETE_APP_EXCEPTION procedure removes a common user or a common user's package from the Database Vault operations control exception list.

- DISABLE_APP_PROTECTION Procedure
  The DISABLE_APP_PROTECTION procedure disables Database Vault operations control.
- DISABLE_DV Procedure
  The DISABLE_DV procedure disables Oracle Database Vault.
- DISABLE_DV_DICTIONARY_ACCTS Procedure
  The DISABLE_DV_DICTIONARY_ACCTS procedure prevents any user from logging into the database as the DVSYS or DVF schema user.
- DISABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The DISABLE_DV_PATCH_ADMIN_AUDIT procedure disables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.
- DISABLE_ORADEBUG Procedure
  The DISABLE_ORADEBUG procedure disables the use of the ORADEBUG utility in an Oracle Database Vault environment.
- ENABLE_APP_PROTECTION Procedure
  The ENABLE_APP_PROTECTION procedure enables Database Vault operations control.
- ENABLE_DV Procedure
  The ENABLE_DV procedure enables Oracle Database Vault and Oracle Label Security.
- ENABLE_DV_DICTIONARY_ACCTS Procedure
  The ENABLE_DV_DICTIONARY_ACCTS procedure enables users to log into the database as the DVSYS or DVF user.
- ENABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The ENABLE_DV_PATCH_ADMIN_AUDIT procedure enables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.
- ENABLE_ORADEBUG Procedure
  The ENABLE_ORADEBUG procedure enables the use of the ORADEBUG utility in an Oracle Database Vault environment.
- UNAUTH_DATAPUMP_CREATE_USER Procedure
  The UNAUTH_DATAPUMP_CREATE_USER procedure removes authorization from an Oracle Data Pump user to create users during an Oracle Data Pump import operation.
- UNAUTH_DATAPUMP_GRANT Procedure
  The UNAUTH_DATAPUMP_GRANT procedure removes authorization from an Oracle Data Pump user to grant Oracle Database Vault-protected roles and system privileges during an Oracle Data Pump import operation.
- UNAUTH_DATAPUMP_GRANT_ROLE Procedure
  The UNAUTH_DATAPUMP_GRANT_ROLE procedure removes authorization from an Oracle Data Pump user to grant a specific role during an Oracle Data Pump import operation.
- UNAUTH_DATAPUMP_GRANT_SYSPRIV Procedure
  The UNAUTH_DATAPUMP_GRANT_SYSPRIV procedure removes authorization from an Oracle Data Pump user to grant system privileges during an Oracle Data Pump import operation.
- UNAUTHORIZE_ADMIN_USER Procedure
  The UNAUTHORIZE_AUDIT_ADMIN procedure revokes the AUDIT_ADMIN authorization from a user.
- UNAUTHORIZE_AUDIT_VIEWER Procedure
  The UNAUTHORIZE_AUDIT_VIEWER procedure revokes the AUDIT_VIEWER authorization from a user.

- UNAUTHORIZE_DATAPUMP_USER Procedure
  The `UNAUTHORIZE_DATAPUMP_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_DATAPUMP_USER` procedure.
- UNAUTHORIZE_DBCAPTURE Procedure
  The `UNAUTHORIZE_DBCAPTURE` procedure revokes authorization from users to perform Oracle Database Replay workload capture operations.
- UNAUTHORIZE_DBREPLAY Procedure
  The `UNAUTHORIZE_DBREPLAY` procedure revokes authorization from users to perform Oracle Database Replay workload replay operations.
- UNAUTHORIZE_DDL Procedure
  The `UNAUTHORIZE_DDL` procedure revokes authorization from a user who was granted authorization to run DDL statements through the `DBMS_MACDM.AUTHORIZE_DDL` procedure.
- UNAUTHORIZE_DIAGNOSTIC_ADMIN Procedure
  The `UNAUTHORIZE_DIAGNOSTIC_ADMIN` procedure revokes authorization from a user who was authorized with the `DBMS_MACADM.AUTHORIZE_DIAGNOSTIC_ADMIN` procedure to query diagnostic views and tables.
- UNAUTHORIZE_MAINTENANCE_USER Procedure
  The `UNAUTHORIZE_MAINTENANCE_USER` procedure revokes privileges from users who have been granted authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.
- UNAUTHORIZE_PREPROCESSOR Procedure
  The `UNAUTHORIZE_PREPROCESSOR` procedure revokes authorization from a user to run preprocessor programs through external tables.
- UNAUTHORIZE_PROXY_USER Procedure
  The `UNAUTHORIZE_PROXY_USER` procedure revokes authorization from a user who was granted proxy authorization from the `DBMS_MACADM.AUTHORIZE_PROXY_USER` procedure.
- UNAUTHORIZE_SCHEDULER_USER Procedure
  The `UNAUTHORIZE_SCHEDULER_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_SCHEDULER_USER` procedure.
- UNAUTHORIZE_SQL_FIREWALL Procedure
  The `UNAUTHORIZE_SQL_FIREWALL` procedure revokes a user's authorization to use SQL Firewall.
- UNAUTHORIZE_TTS_USER Procedure
  The `UNAUTHORIZE_TTS_USER` procedure removes from authorization users who had previously been granted the authorization to perform Oracle Data Pump transportable tablespace operations.

21.1.1 ADD_APP_EXCEPTION Procedure

The `ADD_APP_EXCEPTION` procedure enables a common user or package to access local schemas.

Use this procedure when you are configuring Database Vault operations control to automatically restrict common users from accessing pluggable database (PDB) local data. The procedure applies to the entire container, so you must run it from the CDB root. When the exception is for a package, then owner statements from the given package can access local schemas.

Syntax

```
DBMS_MACADM.ADD_APP_EXCEPTION(
  owner         IN VARCHAR2,
  package_name  IN VARCHAR2);
```

Parameters

Table 21-1 ADD_APP_EXCEPTION

Parameter	Description
owner	Name of the user who you want to add as an exception To find a list of available common users, query the USERNAME and COMMON columns of the DBA_USERS data dictionary view.
package_name	Name of the package that you want to add as an exception if you want to specify a package instead of the entire user account. This package must be owned by the user specified in the owner parameter. If you want to create an exception for the entire schema and not any particular package, then specify '%' for the package_name parameter.

Examples

```
EXEC DBMS_MACADM.ADD_APP_EXCEPTION ('C##HR_ADMIN', '%'); --Applies to the user
c##hr_admin

EXEC DBMS_MACADM.ADD_APP_EXCEPTION('C##HR_ADMIN', 'validateHRdata'); --Applies to the
package validateHRdata
```

Related Topics

- Adding Common Users and Packages to an Exception List
  Common users and applications that must access PDB local data can be added to an exception list.
- ENABLE_APP_PROTECTION Procedure
  The ENABLE_APP_PROTECTION procedure enables Database Vault operations control.
- DISABLE_APP_PROTECTION Procedure
  The DISABLE_APP_PROTECTION procedure disables Database Vault operations control.
- DELETE_APP_EXCEPTION Procedure
  The DELETE_APP_EXCEPTION procedure removes a common user or a common user's package from the Database Vault operations control exception list.

21.1.2 ADD_NLS_DATA Procedure

The ADD_NLS_DATA procedure adds a new language to Oracle Database Vault.

Syntax

```
DBMS_MACADM.ADD_NLS_DATA(
  language      IN VARCHAR );
```

Parameters

Table 21-2 ADD_NLS_DATA

Parameter	Description
language	Enter one of the following settings. (This parameter is case insensitive.) • ENGLISH • GERMAN • SPANISH • FRENCH • ITALIAN • JAPANESE • KOREAN • BRAZILIAN PORTUGUESE • SIMPLIFIED CHINESE • TRADITIONAL CHINESE

Examples

```
EXEC DBMS_MACADM.ADD_NLS_DATA('french');
```

21.1.3 ALLOW_COMMON_OPERATION Procedure

The ALLOW_COMMON_OPERATION procedure controls the access that a local user has on common objects in a PDB.

This procedure can only be run in the CDB root by a common user who has been granted the DV_OWNER role in the root.

Syntax

```
DBMS_MACADM.ALLOW_COMMON_OPERATION(
  status    IN BOOLEAN DEFAULT TRUE);
```

Parameters

Table 21-3 ALLOW_COMMON_OPERATION

Parameter	Description
status	Enter one of the following settings: • TRUE prevents local users from creating Oracle Database Vault controls on common user objects. This setting applies to existing local PDB Database Vault controls that were created on common user objects, so that they will not be enforced on common users. Alternatively, you can run this procedure without including any parameter to achieve a TRUE result. • FALSE enables local users to create Database Vault controls on common user objects. Existing local PDB controls that were created on common user objects will continue to be enforced. If you do not run DBMS_MACADM.ALLOW_COMMON_OPERATION at all, then the default ALLOW COMMON OPERATION status is FALSE, and the default behavior will be to allow local users to create Database Vault controls on common user objects

Example

```
EXEC DBMS_MACADM.ALLOW_COMMON_OPERATION('TRUE');
```

21.1.4 AUTH_DATAPUMP_CREATE_USER Procedure

The AUTH_DATAPUMP_CREATE_USER procedure authorizes an Oracle Data Pump user to create users during an Oracle Data Pump import operation.

This procedure applies to the impdp utility only.

Syntax

```
DBMS_MACADM.AUTH_DATAPUMP_CREATE_USER(
  uname       IN VARCHAR2);
```

Parameters

Table 21-4 AUTH_DATAPUMP_CREATE_USER

Parameter	Description
uname	Name of the Oracle Data Pump user who will need to create users during the import operation. To find the user's current status, query the DBA_DV_DATAPUMP_AUTH data dictionary view.

Example

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

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.5 AUTH_DATAPUMP_GRANT Procedure

The `AUTH_DATAPUMP_GRANT` procedure authorizes an Oracle Data Pump user to grant Oracle Database Vault-protected roles and system privileges during an Oracle Data Pump import operation.

This procedure applies to the `impdp` utility only. Be aware that this authorization does not cover Oracle Database Vault roles such as `DV_OWNER`, `DV_ADMIN`, `DV_MONITOR`, and so on.

Syntax

```
DBMS_MACADM.AUTH_DATAPUMP_GRANT(
  uname       IN VARCHAR2);
```

Parameters

Table 21-5 AUTH_DATAPUMP_GRANT

Parameter	Description
`user_name`	Name of the Oracle Data Pump user who will need to grant roles and privileges to users during the import operation. To find a user's current status, query the `DBA_DV_DATAPUMP_AUTH` data dictionary view.

Example

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

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.6 AUTH_DATAPUMP_GRANT_ROLE Procedure

The `AUTH_DATAPUMP_GRANT_ROLE` procedure authorizes an Oracle Data Pump user to grant a specific role during an Oracle Data Pump import operation.

This procedure applies to the `impdp` utility only.

Syntax

```
DBMS_MACADM.AUTH_DATAPUMP_GRANT_ROLE(
  uname       IN VARCHAR2,
  role        IN VARCHAR2 DEFAULT %);
```

Parameters

Table 21-6 AUTH_DATAPUMP_GRANT_ROLE

Parameter	Description
uname	Name of the Oracle Data Pump user who will need to grant a specific role to users during the import operation. To find a user's current status, query the DBA_DV_DATAPUMP_AUTH data dictionary view.
role	The role to grant to the user. Do not specify Oracle Database Vault roles such as DV_OWNER, DV_ADMIN, DV_MONITOR, and so on. If you omit this value or specify %, then the user is authorized to grant any roles (other than Oracle Database Vault roles) during the import operation. Note that if the user has been authorized with the DBMS_MACADM.AUTH_DATAPUMP_GRANT procedure, or if the user has authorization to grant a specific role, then the user can still grant these roles.

Example

```
EXEC DBMS_MACADM.AUTH_DATAPUMP_GRANT_ROLE('DP_MGR', 'DBA');
```

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.7 AUTH_DATAPUMP_GRANT_SYSPRIV Procedure

The AUTH_DATAPUMP_GRANT_SYSPRIV procedure authorizes an Oracle Data Pump user to grant system privileges during an Oracle Data Pump import operation.

The procedure applies the IMPDP utility only.

Syntax

```
DBMS_MACADM.AUTH_DATAPUMP_GRANT_SYSPRIV(
  uname       IN VARCHAR2);
```

Parameters

Table 21-7 AUTH_DATAPUMP_GRANT_SYSPRIV

Parameter	Description
uname	Name of the Oracle Data Pump user who will need to grant system privileges to users during the IMPDP operation. To find a user's current status, query the DBA_DV_DATAPUMP_AUTH data dictionary view.

Example

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

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.8 AUTHORIZE_AUDIT_ADMIN Procedure

The `AUTHORIZE_AUDIT_ADMIN` procedure enables a user to perform the auditing tasks that are available to users who have been granted the Oracle Database `AUDIT_ADMIN` role, such as creating audit policies.

Syntax

```
DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN(
  uname          IN VARCHAR2);
```

Parameters

Table 21-8 AUTHORIZE_AUDIT_ADMIN

Parameter	Description
`uname`	Name of the user to grant `AUDIT_ADMIN` authorization To find a list of available users, query the `USERNAME` column of the `DBA_USERS` data dictionary view.

Example

```
EXEC DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN ('PSMITH');
```

Related Topics

- Using Oracle Database Auditing with Oracle Database Vault
  You grant and revoke authorizations to manage and view audit records in an Oracle Database Vault environment.

21.1.9 AUTHORIZE_AUDIT_VIEWER Procedure

The `AUTHORIZE_AUDIT_VIEWER` procedure enables a user to perform the auditing tasks that are available to users who have been granted the `AUDIT_VIEWER` role to query audit-related views.

Syntax

```
DBMS_MACADM.AUTHORIZE_AUDIT_VIEWER(
  uname          IN VARCHAR2);
```

Parameters

Table 21-9 AUTHORIZE_AUDIT_VIEWER

Parameter	Description
uname	Name of the user to grant authorization for viewing audit trail information To find a list of available users, query the USERNAME column of the DBA_USERS data dictionary view.

Example

```
EXEC DBMS_MACADM.AUTHORIZE_AUDIT_VIEWER ('PSMITH');
```

Related Topics

-
- Using Oracle Database Auditing with Oracle Database Vault
  You grant and revoke authorizations to manage and view audit records in an Oracle Database Vault environment.

21.1.10 AUTHORIZE_DATAPUMP_USER Procedure

The AUTHORIZE_DATAPUMP_USER procedure authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled.

It applies to both the expdp and impdp utilities.

Syntax

```
DBMS_MACADM.AUTHORIZE_DATAPUMP_USER(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2 DEFAULT NULL,
  table_name     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-10 AUTHORIZE_DATAPUMP_USER

Parameter	Description
user_name	Name of the Oracle Data Pump user to whom you want to grant authorization. To find a list of users who have privileges to use Oracle Data Pump (that is, the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles), query the DBA_ROLE_PRIVS data dictionary view as follows: `SELECT GRANTEE, GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTED_ROLE LIKE '%FULL%'`
schema_name	Name of the database schema that the Oracle Data Pump user must export or import. If you omit this parameter, then the user is granted global authorization to export and import any schema in the database. In this case, ensure the user has been granted the DV_OWNER role.

Table 21-10 (Cont.) AUTHORIZE_DATAPUMP_USER

Parameter	Description
table_name	Name of the table within the schema specified by the schema_name parameter. If you omit this parameter, then the user you specified can export and import all tables within the schema specified by the schema_name parameter.

Examples

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

EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR');

EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('DP_MGR', 'HR', 'EMPLOYEES');
```

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.11 AUTHORIZE_DBCAPTURE Procedure

The AUTHORIZE_DBCAPTURE procedure grants a user authorization to perform Oracle Database Replay workload capture operations.

To find information about users who have been granted this authorization, query the DBA_DV_DBCAPTURE_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_DBCAPTURE(
  uname       IN VARCHAR2);
```

Parameters

Table 21-11 AUTHORIZE_DBCAPTURE

Parameter	Description
uname	Name of the user to whom you want to grant Database Replay workload capture authorization

Example 21-1 Example

```
EXEC DBMS_MACADM.AUTHORIZE_DBCAPTURE('PFITCH');
```

21.1.12 AUTHORIZE_DBREPLAY Procedure

The AUTHORIZE_DBREPLAY procedure grants a user authorization to perform Oracle Database Replay workload replay operations.

To find information about users who have been granted this authorization, query the DBA_DV_DBREPLAY_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_DBREPLAY(
  uname       IN VARCHAR2);
```

Parameters

Table 21-12 AUTHORIZE_DBREPLAY

Parameter	Description
uname	Name of the user to whom you want to grant Database Replay workload replay authorization

Example 21-2 Example

```
EXEC DBMS_MACADM.AUTHORIZE_DBREPLAY('PFITCH');
```

21.1.13 AUTHORIZE_DDL Procedure

The AUTHORIZE_DDL procedure grants a user authorization to run Data Definition Language (DDL) statements on the specified schema.

The DDL authorization allows the grantee to perform DDL operations on users who are authorized to realms or granted Oracle Database Vault roles. However, the DDL authorization does not allow the grantee to perform DDL operations on realm-protected schemas. To enable such operations, you must authorize the user for the realm.

To find information about users who have been granted this authorization, query the DBA_DV_DDL_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_DDL(
  user_name       IN VARCHAR2,
  schema_name     IN VARCHAR2);
```

Parameters

Table 21-13 AUTHORIZE_DDL

Parameter	Description
user_name	Name of the user to whom you want to grant DDL authorization.
schema_name	Name of the database schema in which the user wants to perform the DDL statements. Enter % to specify all schemas.

Examples

The following example enables user psmith to run DDL statements in any schema:

```
EXEC DBMS_MACADM.AUTHORIZE_DDL('psmith', '%');
```

This example enables user psmith to run DDL statements in the HR schema only.

```
EXEC DBMS_MACADM.AUTHORIZE_DDL('psmith', 'HR');
```

Related Topics

- Performing DDL Operations in Oracle Database Vault
  Data Definition Language (DDL) operations in Oracle Database Vault can be affected by situations such as schema ownership and patch upgrades.

21.1.14 AUTHORIZE_DIAGNOSTIC_ADMIN Procedure

The `AUTHORIZE_DIAGNOSTIC_ADMIN` procedure authorizes a user to query diagnostic views and tables.

These views and tables are as follows:

Views and Tables V$	Views and Tables X$
V$DIAG_OPT_TRACE_RECORDS	X$DBGTFOPTT
V$DIAG_SESS_OPT_TRACE_RECORDS	X$DBGTFSOPTT
V$DIAG_TRACE_FILE_CONTENTS	X$DBGTFVIEW

Without this authorization, when a user queries these tables and views, no values are returned.

Syntax

```
DBMS_MACADM.AUTHORIZE_DIAGNOSTIC_ADMIN(
  uname       IN VARCHAR2);
```

Parameters

Table 21-14 AUTHORIZE_DIAGNOSTIC_ADMIN

Parameter	Description
uname	Name of the user to whom you want to grant authorization.

Example

```
EXEC DBMS_MACADM.AUTHORIZE_DIAGNOSTIC_ADMIN('PFITCH');
```

21.1.15 AUTHORIZE_MAINTENANCE_USER Procedure

The `AUTHORIZE_MAINTENANCE_USER` procedure grants a user authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

To find information about users who have been granted this authorization, query the `DBA_DV_MAINTENANCE_AUTH` view.

Syntax

```
DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER(
 uname         IN VARCHAR2,
 sname         IN VARCHAR2 DEFAULT NULL,
 objname       IN VARCHAR2 DEFAULT NULL,
 objtype       IN VARCHAR2 DEFAULT NULL,
 action        IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-15 AUTHORIZE_MAINTENANCE_USER

Parameter	Description
uname	Name of the user to whom you want to grant authorization
sname	Name of the database schema for which the maintenance operations are to be performed. Enter % to specify all schemas.
objname	Name of the object (such as the name of a table) in the schema that is specified in the sname parameter for which maintenance operations are to be performed
objtype	Type of the objname object, such as table, index, tablespace, and so on
action	Maintenance action. Enter ilm for Information Lifecycle Management

Example

The following example enables user psmith to have Database Vault authorization to manage ILM features for the HR.EMPLOYEES table:

```
BEGIN
 DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER (
  uname       => 'psmith',
  sname       => 'HR',
  objname     => 'EMPLOYEES',
  objtype     => 'TABLE',
  action      => 'ILM');
END;
/
```

Related Topics

- Using Information Lifecycle Management with Oracle Database Vault
  Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.

21.1.16 AUTHORIZE_PREPROCESSOR Procedure

The AUTHORIZE_PREPROCESSOR procedure grants a user authorization to run preprocessor programs through external tables.

To find information about users who have been granted this authorization, query the DBA_DV_PREPROCESSOR_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_PREPROCESSOR(
  uname      IN VARCHAR2);
```

Parameters

Table 21-16 AUTHORIZE_PREPROCESSOR

Parameter	Description
uname	Name of the user to whom you want to grant authorization to run preprocessor programs through external tables

Example 21-3 Example

```
EXEC DBMS_MACADM.AUTHORIZE_PREPROCESSOR('PFITCH');
```

Related Topics

- Running Preprocessor Programs with Oracle Database Vault
  Users who run preprocessor programs through external tables must have Oracle Database Vault-specific authorization.
- DBA_DV_PREPROCESSOR_AUTH View
  The DBA_DV_PREPROCESSOR_AUTH data dictionary view shows users who have been granted authorization to run preprocessor programs through external tables.

21.1.17 AUTHORIZE_PROXY_USER Procedure

The AUTHORIZE_PROXY_USER procedure grants a proxy user authorization to proxy other user accounts, as long as the proxy user has database authorization.

For example, the CREATE SESSION privilege is a valid database authorization.

AUTHORIZE_PROXY_USER does not control whether a particular user can connect as a proxy of another user. That part is controlled by GRANT CONNECT THROUGH, which can be issued only by the a user who has the DV_ACCTMGR role. Instead, AUTHORIZE_PROXY_USER controls whether the proxy user is allowed to assume all the Database Vault authorizations that the target user has. For example, suppose that the proxy user hr_proxy_user successfully connects as user HR. Now being HR, hr_proxy_user can access all the objects to which HR has access. However, if the target objects are Database Vault protected and HR is authorized to access it, hr_proxy_user can access the objects if and only if hr_proxy_user is proxy-authorized for HR. If hr_proxy_user is not proxy-authorized for HR, then even after connecting as HR, hr_proxy_user cannot access the Database Vault-protected objects for which HR is authorized.

To find information about users who have been granted authorization using AUTHORIZE_PROXY_USER, query the DBA_DV_PROXY_AUTH view.

Syntax

```
DBMS_MACADM.AUTHORIZE_PROXY_USER(
  proxy_user    IN VARCHAR2,
  user_name     IN VARCHAR2);
```

Parameters

Table 21-17 AUTHORIZE_PROXY_USER

Parameter	Description
proxy_user	Name of the proxy user.
user_name	Name of the database user who will be proxied by the proxy_user user. Enter % to specify all users.

Examples

The following example enables proxy user preston to proxy all users:

```
EXEC DBMS_MACADM.AUTHORIZE_PROXY_USER('preston', '%');
```

This example enables proxy user preston to proxy database user dkent only.

```
EXEC DBMS_MACADM.AUTHORIZE_PROXY_USER('preston', 'dkent');
```

21.1.18 AUTHORIZE_SCHEDULER_USER Procedure

The AUTHORIZE_SCHEDULER_USER procedure grants a user authorization to schedule database jobs when Oracle Database Vault is enabled.

This authorization applies to anyone who has privileges to schedule database jobs. These privileges include any of the following: CREATE JOB, CREATE ANY JOB, CREATE EXTERNAL JOB, EXECUTE ANY PROGRAM, EXECUTE ANY CLASS, MANAGE SCHEDULER.

Syntax

```
DBMS_MACADM.AUTHORIZE_SCHEDULER_USER(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-18 AUTHORIZE_SCHEDULER_USER

Parameter	Description
user_name	Name of the user to whom you want to grant authorization. To find a list of users who have privileges (for example, CREATE JOB and CREATE ANY JOB) to schedule jobs, query the GRANTEE and PRIVILEGE columns of the DBA_SYS_PRIVS data dictionary view.
schema_name	Name of the database schema for which a job will be scheduled. If you omit this parameter, then the user is granted global authorization to schedule a job for any schema in the database.

Examples

The following example authorizes the user JOB_MGR to run a job under any schema.

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR');
```

This example authorizes user JOB_MGR to run a job under the HR schema only.

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

Related Topics

- Using Oracle Scheduler with Oracle Database Vault
  Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific authorization.

21.1.19 AUTHORIZE_SQL_FIREWALL Procedure

The AUTHORIZE_SQL_FIREWALL procedure grants a user authorization to use SQL Firewall.

To find information about users who have been granted this authorization, query the DBA_DV_SQL_FIREWALL_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.AUTHORIZE_SQL_FIREWALL(
  uname              IN VARCHAR2,
  manage_dv_admins   IN VARCHAR2);
```

Parameters

Table 21-19 AUTHORIZE_SQL_FIREWALL

Parameter	Description
uname	Name of the database user to grant SQL Firewall authorization To find a list of available users, query the USERNAME column of the DBA_USERS data dictionary view. The user must have the ADMINISTER SQL FIREWALL system privilege or the SQL_FIREWALL_ADMIN role. You can confirm these by querying the DBA_SYS_PRIVS and DBA_ROLES views.
manage_dv_admins	Specifies whether the authorized user is allowed to apply SQL Firewall policies on user accounts that have been granted the DV_OWNER or DV_ACCTMGR role. The default is N. Specify one of the following settings: • N prevents the user from applying SQL Firewall policies on a target user that is granted the DV_OWNER or DV_ACCTMGR role. • Y enables the user to apply SQL Firewall policies on users who have been granted either the DV_OWNER or DV_ACCTMGR roles. This parameter prevents SQL Firewall from blocking users with DV_OWNER and DV_ACCTMGR (unintentionally or maliciously).

Example

```
EXEC DBMS_MACADM.AUTHORIZE_SQL_FIREWALL('PFITCH', 'Y');
```

Related Topics

- Using SQL Firewall with Oracle Database Vault
  You can authorize SQL Firewall users to work in a Database Vault environment.

21.1.20 AUTHORIZE_TTS_USER Procedure

The AUTHORIZE_TTS_USER procedure authorizes a user to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled.

It applies to both the EXPDP and IMPDP utilities.

Syntax

```
DBMS_MACADM.AUTHORIZE_TTS_USER(
  uname      IN VARCHAR2,
  tsname     IN VARCHAR2);
```

Parameters

Table 21-20 AUTHORIZE_TTS_USER

Parameter	Description
uname	Name of the user who you want to authorize to perform Oracle Data Pump transportable tablespace operations. To find a list of users and their current privileges, query the DBA_SYS_PRIVS data dictionary view.
tsname	Name of the tablespace in which the uname user is to perform the transportable tablespace operation. To find a list of tablespaces, query the DBA_TABLESPACES data dictionary view.

Example

```
EXEC DBMS_MACADM.AUTHORIZE_TTS_USER('PSMITH', 'HR_TS');
```

Related Topics

- Authorizing Users or Roles for Oracle Data Pump Regular Operations in Database Vault
  You can authorize a database administrator or a role to use Data Pump for regular operations in an Oracle Database Vault environment.

21.1.21 DELETE_APP_EXCEPTION Procedure

The DELETE_APP_EXCEPTION procedure removes a common user or a common user's package from the Database Vault operations control exception list.

The exception list allows a user or package to access local PDB data. Removing a user or package from the exception list will block the user or package from accessing PDB local data.

Syntax

```
DBMS_MACADM.DELETE_APP_EXCEPTION(
  owner         IN VARCHAR2,
  package_name  IN VARCHAR2);
```

Parameters

Table 21-21 DELETE_APP_EXCEPTION

Parameter	Description
owner	Name of the user who you want to remove from being an exception
package_name	Name of the package that you want to remove from being an exception

Examples

```
EXEC DBMS_MACADM.DELETE_APP_EXCEPTION ('C##HR_ADMIN'); --Applies to the user
c##hr_admin

EXEC DBMS_MACADM.DELETE_APP_EXCEPTION('C##HR_ADMIN', 'validateHRdata'); --
Applies to the package validateHRdata
```

Related Topics

- Adding Common Users and Packages to an Exception List
  Common users and applications that must access PDB local data can be added to an exception list.
- ADD_APP_EXCEPTION Procedure
  The ADD_APP_EXCEPTION procedure enables a common user or package to access local schemas.
- ENABLE_APP_PROTECTION Procedure
  The ENABLE_APP_PROTECTION procedure enables Database Vault operations control.
- DISABLE_APP_PROTECTION Procedure
  The DISABLE_APP_PROTECTION procedure disables Database Vault operations control.

21.1.22 DISABLE_APP_PROTECTION Procedure

The DISABLE_APP_PROTECTION procedure disables Database Vault operations control.

Syntax

```
DBMS_MACADM.DISABLE_APP_PROTECTION(
  pdb_name     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-22 DISABLE_APP_PROTECTION

Parameter	Description
pdb_name	Name of the pluggable database (PDB) for which you want to disable Database Vault operations control. If you omit this setting, then it applies to all PDBs in the CDB environment. To find a list of available PDBs, query the DBA_PDBS data dictionary view.

Examples

```
EXEC DBMS_MACADM.DISABLE_APP_PROTECTION; --Applies to all PDBs

EXEC DBMS_MACADM.DISABLE_APP_PROTECTION('hr_pdb'); --Applies to a specific PDB
```

Related Topics

- Disabling Database Vault Operations Control
  To disable Database Vault operations control, use the `DBMS_MACADM.DISABLE_APP_PROTECTION` PL/SQL procedure.

21.1.23 DISABLE_DV Procedure

The `DISABLE_DV` procedure disables Oracle Database Vault.

After you run this procedure, you must restart the database.

Syntax

```
DBMS_MACADM.DISABLE_DV;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.DISABLE_DV;
```

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

21.1.24 DISABLE_DV_DICTIONARY_ACCTS Procedure

The `DISABLE_DV_DICTIONARY_ACCTS` procedure prevents any user from logging into the database as the `DVSYS` or `DVF` schema user.

By default these two accounts are locked. Only a user who has been granted the `DV_OWNER` role can run this procedure. To find the status of whether users can log into `DVSYS` and `DVF`, query the `DBA_DV_DICTIONARY_ACCTS` data dictionary view. For stronger security, run this procedure to better protect the `DVSYS` and `DVF` schemas. The disablement takes place immediately, so you do not need to restart the database after running this procedure.

Syntax

```
DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS;
```

Related Topics

- Archiving and Purging the Oracle Database Vault Audit Trail
  If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.

21.1.25 DISABLE_DV_PATCH_ADMIN_AUDIT Procedure

The `DISABLE_DV_PATCH_ADMIN_AUDIT` procedure disables realm, command rule, and rule set auditing of the actions by users who have the `DV_PATCH_ADMIN` role.

This procedure disables the successful actions of this user, not the failed actions. You should run this procedure after the `DV_PATCH_ADMIN` user has completed database patch operation. To find if auditing is enabled or not, query the `DBA_DV_PATCH_AUDIT` data dictionary view.

Syntax

```
DBMS_MACADM.DISABLE_DV_PATCH_ADMIN_AUDIT;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.DISABLE_DV_PATCH_ADMIN_AUDIT;
```

Related Topics

- DV_PATCH_ADMIN Database Vault Database Patch Role
  The `DV_PATCH_ADMIN` role is used for patching operations.
- ENABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The `ENABLE_DV_PATCH_ADMIN_AUDIT` procedure enables realm, command rule, and rule set auditing of the actions by users who have the `DV_PATCH_ADMIN` role.

21.1.26 DISABLE_ORADEBUG Procedure

The `DISABLE_ORADEBUG` procedure disables the use of the `ORADEBUG` utility in an Oracle Database Vault environment.

The disablement takes place immediately, so you do not need to restart the database after running this procedure. To find the status of whether the `ORADEBUG` utility is available in Database Vault, query the `DVYS.DBA_DV_ORADEBUG` data dictionary view.

Syntax

```
DBMS_MACADM.DISABLE_ORADEBUG;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.DISABLE_ORADEBUG;
```

Related Topics

- Using the ORADEBUG Utility with Oracle Database Vault
  The `ORADEBUG` utility is used primarily by Oracle Support to diagnose problems that may arise with an Oracle database.

21.1.27 ENABLE_APP_PROTECTION Procedure

The `ENABLE_APP_PROTECTION` procedure enables Database Vault operations control.

Syntax

```
DBMS_MACADM.ENABLE_APP_PROTECTION(
  pdb_name     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-23 ENABLE_APP_PROTECTION

Parameter	Description
`pdb_name`	Allows a single PDB to have Database Vault operations control re-enabled after it was disabled. The default is to omit the `pdb_name` setting and then enable operations control across all of the PDBs. To find a list of available PDBs, query the `DBA_PDBS` data dictionary view.

Examples

```
EXEC DBMS_MACADM.ENABLE_APP_PROTECTION; --Applies to all PDBs

EXEC DBMS_MACADM.ENABLE_APP_PROTECTION('hr_pdb'); --Applies to a specific PDB
```

Related Topics

- Enabling Database Vault Operations Control
  To enable Database Vault operations control, use the `DBMS_MACADM.ENABLE_APP_PROTECTION` PL/SQL procedure.

21.1.28 ENABLE_DV Procedure

The `ENABLE_DV` procedure enables Oracle Database Vault and Oracle Label Security.

If you want to run `DBMS_MACADM.ENABLE_DV` in an application container, then you must run it in the application container outside of application actions.

After you run this procedure, you must restart the database.

Syntax

```
DBMS_MACADM.ENABLE_DV(
 strict_mode   IN VARCHAR2 DEFAULT);
```

Parameters

Table 21-24 ENABLE_DV

Parameter	Description
strict_mode	Specifies one of the following modes: • n specifies regular mode, which allows the PDBs to be Database Vault enabled or disabled. (Default) • y specifies strict mode, which puts the PDBs that have not been Database Vault-enabled in restricted mode, until you enable Database Vault in them and then restart the PDB. To apply this setting to all PDBs, run the DBMS_MACADM.ENABLE_DV procedure in the CDB root. To apply it to all PDBs in an application container, run the procedure in the application root.

Examples

The following example enables Oracle Database Vault in regular mode.

```
EXEC DBMS_MACADM.ENABLE_DV;
```

This example enables Oracle Database Vault in strict mode.

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

Related Topics

- Disabling and Enabling Oracle Database Vault
  Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

21.1.29 ENABLE_DV_DICTIONARY_ACCTS Procedure

The ENABLE_DV_DICTIONARY_ACCTS procedure enables users to log into the database as the DVSYS or DVF user.

By default, the DVSYS and DVF accounts are locked.

Only a user who has been granted the DV_OWNER role can run this procedure. To find the status of whether users can log into DVSYS and DVF, query the DBA_DV_DICTIONARY_ACCTS data dictionary view. For stronger security, only run this procedure when you need to better protect the DVSYS and DVF schemas. The enablement takes place immediately, so you do not need to restart the database after running this procedure.

Syntax

```
DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS;
```

Related Topics

- Archiving and Purging the Oracle Database Vault Audit Trail
  If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.

21.1.30 ENABLE_DV_PATCH_ADMIN_AUDIT Procedure

The `ENABLE_DV_PATCH_ADMIN_AUDIT` procedure enables realm, command rule, and rule set auditing of the actions by users who have the `DV_PATCH_ADMIN` role.

This procedure is designed to audit these users' actions during a patch upgrade. To find if this auditing is enabled or not, query the `DBA_DV_PATCH_AUDIT` data dictionary view.

Syntax

```
DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT;
```

Related Topics

- DV_PATCH_ADMIN Database Vault Database Patch Role
  The `DV_PATCH_ADMIN` role is used for patching operations.
- DISABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The `DISABLE_DV_PATCH_ADMIN_AUDIT` procedure disables realm, command rule, and rule set auditing of the actions by users who have the `DV_PATCH_ADMIN` role.

21.1.31 ENABLE_ORADEBUG Procedure

The `ENABLE_ORADEBUG` procedure enables the use of the `ORADEBUG` utility in an Oracle Database Vault environment.

The enablement takes place immediately, so you do not need to restart the database after running this procedure. To find the status of whether the `ORADEBUG` utility is available in Database Vault, query the `DVYS.DBA_DV_ORADEBUG` data dictionary view.

Syntax

```
DBMS_MACADM.ENABLE_ORADEBUG;
```

Parameters

None

Example

```
EXEC DBMS_MACADM.ENABLE_ORADEBUG;
```

Related Topics

- Using the ORADEBUG Utility with Oracle Database Vault
  The `ORADEBUG` utility is used primarily by Oracle Support to diagnose problems that may arise with an Oracle database.

21.1.32 UNAUTH_DATAPUMP_CREATE_USER Procedure

The `UNAUTH_DATAPUMP_CREATE_USER` procedure removes authorization from an Oracle Data Pump user to create users during an Oracle Data Pump import operation.

This procedure applies to the `impdp` utility only.

Syntax

```
DBMS_MACADM.UNAUTH_DATAPUMP_CREATE_USER(
  uname       IN VARCHAR2);
```

Parameters

Table 21-25 UNAUTH_DATAPUMP_CREATE_USER

Parameter	Description
`uname`	Name of the Oracle Data Pump user whose authorization must be removed. To find a user's current status, query the `DBA_DV_DATAPUMP_AUTH` data dictionary view.

Example

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

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.33 UNAUTH_DATAPUMP_GRANT Procedure

The `UNAUTH_DATAPUMP_GRANT` procedure removes authorization from an Oracle Data Pump user to grant Oracle Database Vault-protected roles and system privileges during an Oracle Data Pump import operation.

This procedure applies to the `impdp` utility only.

Syntax

```
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT(
  uname       IN VARCHAR2);
```

Parameters

Table 21-26 UNAUTH_DATAPUMP_GRANT

Parameter	Description
user_name	Name of the Oracle Data Pump user whose authorization must be removed. To find a user's current status, query the DBA_DV_DATAPUMP_AUTH data dictionary view.

Example

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

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.34 UNAUTH_DATAPUMP_GRANT_ROLE Procedure

The UNAUTH_DATAPUMP_GRANT_ROLE procedure removes authorization from an Oracle Data Pump user to grant a specific role during an Oracle Data Pump import operation.

This procedure applies to the impdp utility only.

Syntax

```
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_ROLE(
  uname       IN VARCHAR2,
  role        IN VARCHAR2 DEFAULT %);
```

Parameters

Table 21-27 UNAUTH_DATAPUMP_GRANT_ROLE

Parameter	Description
uname	Name of the Oracle Data Pump user whose authorization must be removed. To find a user's current status, query the DBA_DV_DATAPUMP_AUTH data dictionary view.
role	The role that the user is authorized to grant during the import operation. Do not specify Oracle Database Vault roles such as DV_OWNER, DV_ADMIN, DV_MONITOR, and so on. If you omit this value, then the user is not authorized to grant roles during the import.

Example

```
EXEC DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_ROLE('DP_MGR', 'DBA');
```

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.35 UNAUTH_DATAPUMP_GRANT_SYSPRIV Procedure

The UNAUTH_DATAPUMP_GRANT_SYSPRIV procedure removes authorization from an Oracle Data Pump user to grant system privileges during an Oracle Data Pump import operation.

This procedure applies the impdp utility only.

Syntax

```
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_SYSPRIV(
  uname       IN VARCHAR2);
```

Parameters

Table 21-28 UNAUTH_DATAPUMP_GRANT_SYSPRIV

Parameter	Description
uname	Name of the Oracle Data Pump user whose authorization must be removed. To find a user's current status, query the DBA_DV_DATAPUMP_AUTH data dictionary view.

Example

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

Related Topics

- Authorizing Users or Roles for Data Pump Regular Export and Import Operations
  You can use different authorization types for administrators who perform Oracle Data Pump export and import operations in a Database Vault environment.

21.1.36 UNAUTHORIZE_ADMIN_USER Procedure

The UNAUTHORIZE_AUDIT_ADMIN procedure revokes the AUDIT_ADMIN authorization from a user.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_AUDIT_ADMIN(
  uname          IN VARCHAR2);
```

Parameters

Table 21-29 UNAUTHORIZE_AUDIT_ADMIN

Parameter	Description
uname	Name of the user to revoke AUDIT_ADMIN authorization from To find users who have been granted AUDIT_ADMIN authorization, query the DBA_DV_AUDIT_ADMIN_AUTH data dictionary view.

Examples

```
EXEC DBMS_MACADM.UNAUTHORIZE_AUDIT_ADMIN ('PSMITH');
```

Related Topics

- Using Oracle Database Auditing with Oracle Database Vault
  You grant and revoke authorizations to manage and view audit records in an Oracle Database Vault environment.

21.1.37 UNAUTHORIZE_AUDIT_VIEWER Procedure

The `UNAUTHORIZE_AUDIT_VIEWER` procedure revokes the `AUDIT_VIEWER` authorization from a user.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_AUDIT_VIEWER(
  uname         IN VARCHAR2;
```

Parameters

Table 21-30 UNAUTHORIZE_AUDIT_VIEWER

Parameter	Description
`uname`	Name of the user to revoke `AUDIT_VIEWER` authorization from To find users who have been granted `AUDIT_VIEWER` authorization, query the `DBA_DV_AUDIT_VIEWER_AUTH` data dictionary view.

Examples

```
EXEC DBMS_MACADM.UNAUTHORIZE_AUDIT_VIEWER ('RLAYTON');
```

Related Topics

- Using Oracle Database Auditing with Oracle Database Vault
  You grant and revoke authorizations to manage and view audit records in an Oracle Database Vault environment.

21.1.38 UNAUTHORIZE_DATAPUMP_USER Procedure

The `UNAUTHORIZE_DATAPUMP_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_DATAPUMP_USER` procedure.

When you run this procedure, ensure that its settings correspond exactly to the equivalent `AUTHORIZE_DATAPUMP_USER` procedure.

For example, the following two procedures will work because the parameters are consistent:

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

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

However, because the parameters in the following procedures are not consistent, the `UNAUTHORIZE_DATAPUMP_USER` procedure will not work:

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

EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('JSMITH', 'HR');
```

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER(
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2 DEFAULT NULL,
  table_name     IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-31 UNAUTHORIZE_DATAPUMP_USER

Parameter	Description
user_name	Name of the Oracle Data Pump user from whom you want to revoke authorization. To find a list of users and authorizations from the AUTHORIZE_DATAPUMP_USER procedure, query the DBA_DV_DATAPUMP_AUTH data dictionary view as follows: `SELECT * FROM DBA_DV_DATAPUMP_AUTH;`
schema_name	Name of the database schema that the Oracle Data Pump user is authorized to export or import.
table_name	Name of the table within the schema specified by the schema name parameter.

Examples

```
EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('JSMITH');

EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('JSMITH', 'HR');

EXEC DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER('JSMITH', 'HR', 'SALARY');
```

21.1.39 UNAUTHORIZE_DBCAPTURE Procedure

The UNAUTHORIZE_DBCAPTURE procedure revokes authorization from users to perform Oracle Database Replay workload capture operations.

To find information about users who have been granted this authorization, query the DBA_DV_DBCAPTURE_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DBCAPTURE(
  uname      IN VARCHAR2);
```

Parameters

Table 21-32 UNAUTHORIZE_DBCAPTURE

Parameter	Description
uname	Name of the user from whom you want to revoke Database Replay workload capture authorization

Example 21-4 Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_DBCAPTURE('PFITCH');
```

21.1.40 UNAUTHORIZE_DBREPLAY Procedure

The UNAUTHORIZE_DBREPLAY procedure revokes authorization from users to perform Oracle Database Replay workload replay operations.

To find information about users who have been granted this authorization, query the DBA_DV_DBREPLAY_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DBREPLAY(
  uname       IN VARCHAR2);
```

Parameters

Table 21-33 UNAUTHORIZE_DBREPLAY

Parameter	Description
uname	Name of the user from whom you want to revoke Database Replay workload replay authorization

Example 21-5 Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_DBREPLAY('PFITCH');
```

21.1.41 UNAUTHORIZE_DDL Procedure

The UNAUTHORIZE_DDL procedure revokes authorization from a user who was granted authorization to run DDL statements through the DBMS_MACDM.AUTHORIZE_DDL procedure.

To find information about users who have been granted this authorization, query the DBA_DV_DDL_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DDL(
  user_name       IN VARCHAR2,
  schema_name     IN VARCHAR2);
```

Parameters

Table 21-34 UNAUTHORIZE_DDL

Parameter	Description
user_name	Name of the user from whom you want to revoke DDL authorization.
schema_name	Name of the database schema in which the user wants to perform the DDL statements. Enter % specify all schemas.

Examples

The following example revokes DDL statement execution authorization from user psmith for all schemas:

```
EXEC DBMS_MACADM.UNAUTHORIZE_DDL('psmith', '%');
```

This example revokes DDL statement execution authorization from user psmith for the HR schema only.

```
EXEC DBMS_MACADM.UNAUTHORIZE_DDL('psmith', 'HR');
```

Related Topics

- Performing DDL Operations in Oracle Database Vault
  Data Definition Language (DDL) operations in Oracle Database Vault can be affected by situations such as schema ownership and patch upgrades.

21.1.42 UNAUTHORIZE_DIAGNOSTIC_ADMIN Procedure

The UNAUTHORIZE_DIAGNOSTIC_ADMIN procedure revokes authorization from a user who was authorized with the DBMS_MACADM.AUTHORIZE_DIAGNOSTIC_ADMIN procedure to query diagnostic views and tables.

These views and tables are as follows:

Views and Tables V$	Views and Tables X$
V$DIAG_OPT_TRACE_RECORDS	X$DBGTFOPTT
V$DIAG_SESS_OPT_TRACE_RECORDS	X$DBGTFSOPTT
V$DIAG_TRACE_FILE_CONTENTS	X$DBGTFVIEW

Without this authorization, when a user queries these tables and views, no values are returned.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_DIAGNOSTIC_ADMIN(
  uname      IN VARCHAR2);
```

Parameters

Table 21-35 UNAUTHORIZE_DIAGNOSTIC_ADMIN

Parameter	Description
uname	Name of the user from whom you want to revoke authorization.

Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_DIAGNOSTIC_ADMIN('PFITCH');
```

21.1.43 UNAUTHORIZE_MAINTENANCE_USER Procedure

The UNAUTHORIZE_MAINTENANCE_USER procedure revokes privileges from users who have been granted authorization to perform Information Lifecycle Management (ILM) operations in an Oracle Database Vault environment.

To find information about the settings for the ILM authorization, query the DBA_DV_MAINTENANCE_AUTH view.

When you run this procedure, ensure that its settings correspond exactly to the equivalent AUTHORIZE_MAINTENANCE_USER procedure.

For example, the following two procedures will work because the parameter settings correspond:

```
EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER('psmith', 'OE', 'ORDERS', 'TABLE', 'ILM');
EXEC DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER('psmith', 'OE', 'ORDERS', 'TABLE',
'ILM');
```

However, these two statements will fail because the settings do not correspond:

```
EXEC DBMS_MACADM.AUTHORIZE_MAINTENANCE_USER('psmith', 'OE', 'ORDERS', 'TABLE', 'ILM');

EXEC DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER('psmith', '%', '%', '%', 'ILM');
```

Syntax

```
DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER(
 uname         IN VARCHAR2,
 sname         IN VARCHAR2 DEFAULT NULL,
 objname       IN VARCHAR2 DEFAULT NULL,
 objtype       IN VARCHAR2 DEFAULT NULL,
 action        IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-36 UNAUTHORIZE_MAINTENANCE_USER

Parameter	Description
uname	Name of the user from whom you want to revoke authorization
sname	Name of the database schema for which the maintenance operations are performed. Enter % to specify all schemas.

Table 21-36 (Cont.) UNAUTHORIZE_MAINTENANCE_USER

Parameter	Description
objname	Name of the object (such as the name of a table) in the schema that is specified in the sname parameter for which maintenance operations are performed
objtype	Type of the objname object, such as table, index, tablespace, and so on
action	Maintenance action. Enter ilm for Information Lifecycle Management

Example

The following example revokes privileges from Database Vault user psmith so that they can no longer perform ILM operations in any HR schema objects:

```
BEGIN
 DBMS_MACADM.UNAUTHORIZE_MAINTENANCE_USER (
  uname       => 'psmith',
  sname       => 'HR',
  objname     => 'EMPLOYEES',
  objtype     => 'TABLE',
  action      => 'ILM');
END;
/
```

Related Topics

- Using Information Lifecycle Management with Oracle Database Vault
  Users who perform Information Lifecycle Management operations on an Oracle Database Vault-enabled database must be granted authorization to perform these operations.

21.1.44 UNAUTHORIZE_PREPROCESSOR Procedure

The UNAUTHORIZE_PREPROCESSOR procedure revokes authorization from a user to run preprocessor programs through external tables.

To find information about users who have been granted this authorization, query the DBA_DV_PREPROCESSOR_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR(
  uname      IN VARCHAR2);
```

Parameters

Table 21-37 UNAUTHORIZE_PREPROCESSOR

Parameter	Description
uname	Name of the user from whom you want to revoke authorization to run preprocessor programs through external tables

Example 21-6 Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR('PFITCH');
```

Related Topics

- Running Preprocessor Programs with Oracle Database Vault
  Users who run preprocessor programs through external tables must have Oracle Database Vault-specific authorization.
- DBA_DV_PREPROCESSOR_AUTH View
  The `DBA_DV_PREPROCESSOR_AUTH` data dictionary view shows users who have been granted authorization to run preprocessor programs through external tables.

21.1.45 UNAUTHORIZE_PROXY_USER Procedure

The `UNAUTHORIZE_PROXY_USER` procedure revokes authorization from a user who was granted proxy authorization from the `DBMS_MACADM.AUTHORIZE_PROXY_USER` procedure.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_PROXY_USER(
  proxy_user   IN VARCHAR2,
  user_name    IN VARCHAR2);
```

Parameters

Table 21-38 UNAUTHORIZE_PROXY_USER

Parameter	Description
`proxy_user`	Name of the proxy user from whom you want to revoke authorization.
`user_name`	Name of the database user who was proxied by the `proxy_user` user. Enter `%` to specify all users.

Examples

The following example revokes proxy authorization from user `preston` for proxying all users:

```
DBMS_MACADM.UNAUTHORIZE_PROXY_USER('preston', '%');
```

This example revokes proxy authorization from user `preston` for proxying database user `psmith` only.

```
EXEC DBMS_MACADM.UNAUTHORIZE_PROXY_USER('preston', 'psmith');
```

21.1.46 UNAUTHORIZE_SCHEDULER_USER Procedure

The `UNAUTHORIZE_SCHEDULER_USER` procedure revokes the authorization that was granted by the `AUTHORIZE_SCHEDULER_USER` procedure.

When you run this procedure, ensure that its settings correspond exactly to the equivalent `AUTHORIZE_SCHEDULER_USER` procedure. For example, the following two procedures will work because the parameters are consistent:

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR');

EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR');
```

However, because the parameters in the following procedures are not consistent, the `UNAUTHORIZE_SCHEDULER_USER` procedure will not work:

```
EXEC DBMS_MACADM.AUTHORIZE_SCHEDULER_USER('JOB_MGR');

EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

Syntax

```
DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER
  user_name      IN VARCHAR2,
  schema_name    IN VARCHAR2 DEFAULT NULL);
```

Parameters

Table 21-39 UNAUTHORIZE_SCHEDULER_USER

Parameter	Description
user_name	Name of the job scheduling user from whom you want to revoke authorization. To find a list of users and authorizations from the AUTHORIZE_SCHEDULER_USER procedure, query the DBA_DV_JOB_AUTH data dictionary view as follows: `SELECT * FROM DBA_DV_JOB_AUTH;`
schema_name	Name of the database schema for which the user is authorized to schedule jobs.

Examples

```
EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR');

EXEC DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER('JOB_MGR', 'HR');
```

21.1.47 UNAUTHORIZE_SQL_FIREWALL Procedure

The UNAUTHORIZE_SQL_FIREWALL procedure revokes a user's authorization to use SQL Firewall.

When you run this procedure, ensure that its settings correspond exactly to the equivalent AUTHORIZE_SQL_FIREWALL procedure.

For example, the following two procedures will work because the parameter settings are consistent:

```
EXEC DBMS_MACADM.AUTHORIZE_SQL_FIREWALL('PFITCH', 'Y');
EXEC DBMS_MACADM.UNAUTHORIZE_SQL_FIREWALL('PFITCH', 'Y');
```

However, because the parameter settings in the following procedures are not consistent, the UNAUTHORIZE_SQL_FIREWALL procedure will not work and will return an ORA-47896: SQL Firewall authorization for Oracle Database Vault to PFITCH is not found error:

```
EXEC DBMS_MACADM.AUTHORIZE_SQL_FIREWALL('PFITCH', 'N');
EXEC DBMS_MACADM.UNAUTHORIZE_SQL_FIREWALL('PFITCH', 'Y');
```

To find information about users who have been granted SQL Firewall authorization, query the DBA_DV_SQL_FIREWALL_AUTH data dictionary view.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_SQL_FIREWALL(
  uname              IN VARCHAR2,
  manage_dv_admins   IN VARCHAR2);
/
```

Parameters

Table 21-40 UNAUTHORIZE_SQL_FIREWALL

Parameter	Description
uname	Name of the database user from which to revoke SQL Firewall authorization To find a list of authorized users, query the GRANTEE column of the DBA_DV_SQL_FIREWALL_AUTH data dictionary view.
manage_dv_admins	Specifies whether the authorized user is allowed to apply SQL Firewall policies on user accounts that have been granted the DV_OWNER or DV_ACCTMGR role. The default is N. Specify one of the following settings: • N prevents the user from applying SQL Firewall policies on a target user that is granted the DV_OWNER or DV_ACCTMGR role. • Y enables the user to apply SQL Firewall policies on users who have been granted either the DV_OWNER or DV_ACCTMGR roles. This parameter prevents SQL Firewall from blocking users with DV_OWNER and DV_ACCTMGR (unintentionally or maliciously).

Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_SQL_FIREWALL('PFITCH', 'Y');
```

Related Topics

- Using SQL Firewall with Oracle Database Vault
  You can authorize SQL Firewall users to work in a Database Vault environment.

21.1.48 UNAUTHORIZE_TTS_USER Procedure

The UNAUTHORIZE_TTS_USER procedure removes from authorization users who had previously been granted the authorization to perform Oracle Data Pump transportable tablespace operations.

Syntax

```
DBMS_MACADM.UNAUTHORIZE_TTS_USER
  uname      IN VARCHAR2,
  tsname     IN VARCHAR2);
```

Parameters

Table 21-41 UNAUTHORIZE_TTS_USER

Parameter	Description
uname	Name of the user who you want to remove from being authorized to perform Oracle Data Pump transportable tablespace operations. To find a list of users and their current privileges, query the DBA_SYS_PRIVS data dictionary view.
tsname	Name of the tablespace that is used in the transportable tablespace operation. To find a list of tablespaces, query the DBA_TABLESPACES data dictionary view.

Example

```
EXEC DBMS_MACADM.UNAUTHORIZE_TTS_USER('PSMITH', 'HR_TS');
```

21.2 CONFIGURE_DV General System Maintenance Procedure

The CONFIGURE_DV procedure configures the initial two Oracle Database user accounts, which are granted the DV_OWNER and DV_ACCTMGR roles, respectively.

You can check the status of this configuration by querying the DBA_DV_STATUS data dictionary view. Before you run the CONFIGURE_DV procedure, you must create the two user accounts and grant them the CREATE SESSION privilege. The accounts can be either local or common. If you create common user accounts, then the Database Vault roles that are granted to these users apply to the current pluggable database (PDB) only. You then refer to these user accounts for the CONFIGURE_DV procedure.

The CONFIGURE_DV procedure resides in the SYS schema. Oracle provides a synonym, DVSYS.CONFIGURE_DV, so that any existing Oracle Database Vault configuration scripts that you may have created in previous releases will continue to work in this release.

You only can run the CONFIGURE_DV procedure once, when you are ready to configure and enable Oracle Database Vault with an Oracle database. After you run this procedure, you must run utlrp.sql script and then DBMS_MACADM.ENABLE_DV to complete the registration process. Oracle strongly recommends that for better security, you use the two accounts you create here as back-up accounts and then create additional accounts for every day use.

If after running CONFIGURE_DV you decide that you want to modify the settings that you had entered, you or another user who has the DV_OWNER role must disable Database Vault, and then have an administrator with the SYSDBA or SYSOPER administrative privilege restart the database. As user SYS, then commonly grant the DV_OWNER user the DV_OWNER role, with the CONTAINER clause set to ALL.

When you run the CONFIGURE_DV procedure, it checks the DVSYS schema for problems such as missing tables or packages. If it finds problems, then it raises an ORA-47500

`Database Vault cannot be configured` error. If this happens, then you can reinstall Oracle Database Vault onto a PDB by running `catmac.sql`.

Together, the `CONFIGURE_DV` and `DBMS_MACADM.ENABLE_DV` procedures, and the and `utlrp.sql` script, are designed to be a command-line alternative to using Oracle Database Configuration Assistant (DBCA) to configure and enable Oracle Database Vault with an Oracle database.

When you configure and enable Oracle Database Vault with an Oracle database, you must run the `CONFIGURE_DV` procedure as user `SYS`.

Syntax

```
CONFIGURE_DV
  dvowner_uname          IN VARCHAR2,
  dvacctmgr_uname        IN VARCHAR2,
  force_local_dvowner    IN BOOLEAN;
```

Parameters

Table 21-42 CONFIGURE_DV

Parameter	Description
`dvowner_uname`	Name of the user who will be the Database Vault Owner. This user will be granted the `DV_OWNER` role.
`dvacctmgr_uname`	Name of the user who will be the Database Vault Account Manager. This user will be granted the `DV_ACCTMGR` role. If you omit this setting, the user specified by the `dvowner_uname` parameter is made the Database Vault Account Manager and granted the `DV_ACCTMGR` role.
`force_local_dvowner`	Applies only to the `DV_OWNER` (`dvowner_unname` user) in the CDB root or an application root. It does not apply to users who are created in a PDB. • `TRUE` restricts the `DV_OWNER` role privileges of the `dvowner_unname` user to be local to the root. • `FALSE`, the default setting,enables the `dvowner_unname` user to have `DV_OWNER` privileges for all containers that are associated with the root.

Example

```
CREATE USER c##dbv_owner_root_backup IDENTIFIED BY password CONTAINER = CURRENT;
CREATE USER c##dbv_acctmgr_root_backup IDENTIFIED BY password CONTAINER = CURRENT;
GRANT CREATE SESSION TO c##dbv_owner_root_backup, c##dbv_acctmgr_root_backup;

BEGIN
 CONFIGURE_DV (
   dvowner_uname          => 'c##dbv_owner_root_backup',
   dvacctmgr_uname        => 'c##adbv_acctmgr_root_backup',
   force_local_dvowner    => TRUE);
 END;
/
```

Related Topics

- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the `DV_OWNER` and `DV_ACCTMGR` roles.

- Uninstalling Oracle Database Vault
  You can uninstall Oracle Database Vault from an Oracle Database installation, for PDBs (but not the root) and Oracle RAC installations.
- Reinstalling Oracle Database Vault
  You can reinstall Oracle Database Vault by manually installing it, and then afterward, configure and enable it.
- Getting Started with Oracle Database Vault
  Before you can start using Oracle Database Vault, you must configure and enable it with the Oracle database.

22

Oracle Database Vault Policy APIs

You can use the `DBMS_MACADM` PL/SQL package to manage Oracle Database Vault policies.

Only users who have been granted the `DV_OWNER` or `DV_ADMIN` role can use these procedures.

- ADD_CMD_RULE_TO_POLICY Procedure
  The `ADD_COMMAND_RULE_TO_POLICY` procedure enables you to add an existing command rule to an Oracle Database Vault policy.
- ADD_OWNER_TO_POLICY Procedure
  The `ADD_OWNER_TO_POLICY` procedure enables you to add an existing database user to an Oracle Database Vault policy as an owner.
- ADD_REALM_TO_POLICY Procedure
  The `ADD_REALM_TO_POLICY` procedure enables you to add an existing realm to an Oracle Database Vault policy.
- CREATE_POLICY Procedure
  The `CREATE_POLICY` procedure enables you to create an Oracle Database Vault policy.
- DELETE_CMD_RULE_FROM_POLICY Procedure
  The `DELETE_CMD_RULE_FROM_POLICY` procedure enables you to remove an existing command rule from an Oracle Database Vault policy.
- DELETE_OWNER_FROM_POLICY Procedure
  The `DELETE_OWNER_FROM_POLICY` procedure enables you to remove an owner from an Oracle Database Vault policy.
- DELETE_REALM_FROM_POLICY Procedure
  The `DELETE_REALM_FROM_POLICY` procedure enables you to remove an existing realm from an Oracle Database Vault policy.
- DROP_POLICY Procedure
  The `DROP_POLICY` procedure enables you to drop an existing Oracle Database Vault policy.
- RENAME_POLICY Procedure
  The `UPDATE_POLICY_DESCRIPTION` procedure enables you to rename an existing Oracle Database Vault policy.
- UPDATE_POLICY_DESCRIPTION Procedure
  The `UPDATE_POLICY_DESCRIPTION` procedure enables you to update the `description` field in an Oracle Database Vault policy.
- UPDATE_POLICY_STATE Procedure
  The `UPDATE_POLICY_STATE` procedure enables you to update the `policy_state` field in an Oracle Database Vault policy.

Related Topics

- Configuring Oracle Database Vault Policies
  You can use Oracle Database Vault policies to implement frequently used realm and command rule settings.

- Oracle Database Vault Utility APIs
  Oracle Database Vault provides a set of utility APIs in the `DBMS_MACUTL` PL/SQL package.

22.1 ADD_CMD_RULE_TO_POLICY Procedure

The `ADD_COMMAND_RULE_TO_POLICY` procedure enables you to add an existing command rule to an Oracle Database Vault policy.

You can add a command rule to a policy when the command rule is in any state. For example, you can add a disabled command rule to an enabled policy. In this case, the disabled command rule will automatically become enabled when it is added to the policy. A command rules can be added to only one policy. In other words, you cannot assign the same command rule to multiple policies.

Syntax

```
DBMS_MACADM.ADD_CMD_RULE_TO_POLICY(
 policy_name    IN VARCHAR2,
 command        IN VARCHAR2,
 object_owner   IN VARCHAR2,
 object_name    IN VARCHAR2,
 clause_name    IN VARCHAR2 DEFAULT,
 parameter_name IN VARCHAR2 DEFAULT,
 event_name     IN VARCHAR2 DEFAULT,
 component_name IN VARCHAR2 DEFAULT,
 action_name    IN VARCHAR2 DEFAULT,
 scope          IN NUMBER DEFAULT);
```

Parameters

Table 22-1 ADD_CMD_RULE_TO_POLICY Parameters

Parameter	Description
`policy_name`	Policy name. To find existing Database Vault policies in the current database instance, query the `DBA_DV_POLICY` view.
`command`	Command rule name To find existing Database Vault command rules in the current database instance, query the `DBA_DV_COMMAND_RULE` view.
`object_owner`	Database schema to which the command rule applies To find existing object owners for this command rule, query the `DBA_DV_COMMAND_RULE` view.
`object_name`	Object to be protected by the command rule To find existing objects for this command rule, query the `DBA_DV_COMMAND_RULE` view.
`clause_name`	For `ALTER SYSTEM` and `ALTER SESSION` command rules, a clause from the SQL statement that was used to create the command rule To find existing clauses for this command rule, query the `DBA_DV_COMMAND_RULE` view.

Table 22-1 (Cont.) ADD_CMD_RULE_TO_POLICY Parameters

Parameter	Description
parameter_name	For ALTER SYSTEM and ALTER SESSION command rules, a parameter from the clause_name parameter. To find existing parameters for this command rule, query the DBA_DV_COMMAND_RULE view.
event_name	For ALTER SYSTEM and ALTER SESSION command rules, an event that the command rule defines To find existing event names for this command rule, query the DBA_DV_COMMAND_RULE view.
component_name	A component of the event_name setting To find existing component names for this command rule, query the DBA_DV_COMMAND_RULE view.
action_name	An action of the component_name setting. To find existing action names for this command rule, query the DBA_DV_COMMAND_RULE view.
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule applies to all the PDBs

Example

The following example shows how to add a common command rule to a Database Vault policy. This command rule is in the application root of a multitenant environment, so the user running this procedure must be in the application root or the CDB root. Any rules or rule sets that are associated with this command rule must be common.

```
BEGIN
 DBMS_MACADM.ADD_CMD_RULE_TO_POLICY(
  policy_name    => 'HR_DV_Policy',
  command        => 'ALTER SESSION',
  object_owner   => '%',
  object_name    => '%',
  clause_name    => 'PARALLEL DDL',
  parameter_name => '',
  event_name     => '',
  action_name    => '',
  scope          => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

22.2 ADD_OWNER_TO_POLICY Procedure

The ADD_OWNER_TO_POLICY procedure enables you to add an existing database user to an Oracle Database Vault policy as an owner.

When you add an owner to an enabled policy, the change takes place immediately. There is no limit to the number of users that you add to the policy.

Syntax

```
DBMS_MACADM.ADD_OWNER_TO_POLICY(
  policy_name    IN VARCHAR2,
  owner_name     IN VARCHAR2);
```

Parameters

Table 22-2 ADD_OWNER_TO_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view.
owner_name	User name. To find existing database users (not roles) in the current instance, query the DBA_USERS view. To find existing policy owners, query the DBA_DV_POLICY_OWNER view.

Example

```
BEGIN
 DBMS_MACADM.ADD_OWNER_TO_POLICY(
  policy_name     => 'HR_DV_Policy',
  owner_name      => 'PSMITH');
END;
/
```

22.3 ADD_REALM_TO_POLICY Procedure

The ADD_REALM_TO_POLICY procedure enables you to add an existing realm to an Oracle Database Vault policy.

You can add a disabled realm to an enabled policy. In this case, the realm automatically becomes enabled when it is added. A realm can be added to only one policy. In other words, you cannot assign the same realm to multiple policies.

Syntax

```
DBMS_MACADM.ADD_REALM_TO_POLICY(
  policy_name    IN VARCHAR2,
  realm_name     IN VARCHAR2);
```

Parameters

Table 22-3 ADD_REALM_TO_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view.
realm_name	Realm name. To find existing Database Vault realms in the current database instance.

Example

```
BEGIN
 DBMS_MACADM.ADD_REALM_TO_POLICY(
  policy_name    => 'HR_DV_Policy',
  realm_name     => 'HR Realm');
END;
/
```

22.4 CREATE_POLICY Procedure

The `CREATE_POLICY` procedure enables you to create an Oracle Database Vault policy.

After you create the policy, you must add at least one realm and one command rule to the policy. Optionally, you can set these realms and command rules to be enforced individually or use the enforcement that the policy uses.

An owner for the policy is not required, but if you do not assign an owner to the policy, a user who has been granted the `DV_OWNER` or `DV_ADMIN` role must administer the policy.

After you create the policy, use the following procedures to complete the policy definition:

- `ADD_REALM_TO_POLICY` adds realms to the policy.
- `ADD_CMD_RULE_TO_POLICY` adds command rules to the policy.
- `ADD_OWNER_TO_POLICY` enables the specified database users to manage the policy.

Syntax

```
DBMS_MACADM.CREATE_POLICY(
 policy_name  IN VARCHAR2,
 description  IN VARCHAR2 DEFAULT,
 policy_state IN NUMBER,
 pl_sql_stack IN BOOLEAN DEFAULT);
```

Parameters

Table 22-4 CREATE_POLICY Parameters

Parameter	Description
`policy_name`	Policy name, up to 128 characters in mixed case To find existing policies in the current database instance, query the `DBA_DV_POLICY` view.
`description`	Description of the purpose of the policy, up to 4000 characters in mixed-case.

Table 22-4 (Cont.) CREATE_POLICY Parameters

Parameter	Description
policy_state	Specifies how the policy is enabled. Possible values are: • DBMS_MACADM.G_ENABLED (1), which enables the policy after you create it • DBMS_MACADM.G_DISABLED (0), which disables the policy after you create it • DBMS_MACADM.G_SIMULATION (2), which sets the policy to simulation mode. In simulation mode, any violations to realms or command rules used in the policy are logged in a designated log table with sufficient information to describe the error, such as the user name or SQL statement used. See also Related Topics. • DBMS_MACADM.G_PARTIAL (3), which sets the policy to partial mode. In partial mode, the enforcement state of realms or command rules associated with the policy can be changed individually.
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Example

The following example creates a policy that uses the partial state and enables the capture of the PL/SQL stack. Later on, when a realm or a command rule is added to this policy, their enforcement state will be able to be changed individually.

```
BEGIN
 DBMS_MACADM.CREATE_POLICY(
  policy_name  => 'HR_DV_Policy',
  description  => 'Policy to protect the HR schema',
  policy_state => DBMS_MACADM.G_ENABLED,
  pl_sql_stack => TRUE);
END;
/
```

Related Topics

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

22.5 DELETE_CMD_RULE_FROM_POLICY Procedure

The DELETE_CMD_RULE_FROM_POLICY procedure enables you to remove an existing command rule from an Oracle Database Vault policy.

You can remove command rules from a policy anytime regardless of the state of the policy. When a command rule is removed from a policy, the state of command rule remains the same. That is, if the policy is enabled, and a command rule is removed from the policy, then the command rule will be still enabled after you have removed it from the policy.

Syntax

```
DBMS_MACADM.DELETE_CMD_RULE_FROM_POLICY(
 policy_name    IN VARCHAR2,
```

```
command        IN VARCHAR2,
object_owner   IN VARCHAR2,
object_name    IN VARCHAR2,
clause_name    IN VARCHAR2 DEFAULT,
parameter_name IN VARCHAR2 DEFAULT,
event_name     IN VARCHAR2 DEFAULT,
component_name IN VARCHAR2 DEFAULT,
action_name    IN VARCHAR2 DEFAULT,
scope          IN NUMBER DEFAULT);
```

Parameters

Table 22-5 DELETE_CMD_RULE_FROM_POLICY Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view.
command	Command rule name To find existing Database Vault command rules in the current database instance, query the DBA_DV_COMMAND_RULE view.
object_owner	Database schema to which the command rule applies To find existing object owners for this command rule, query the DBA_DV_COMMAND_RULE view.
object_name	Object to be protected by the command rule To find existing objects for this command rule, query the DBA_DV_COMMAND_RULE view.
clause_name	For ALTER SYSTEM and ALTER SESSION command rules, a clause from the SQL statement that was used to create the command rule To find existing clauses for this command rule, query the DBA_DV_COMMAND_RULE view.
parameter_name	For ALTER SYSTEM and ALTER SESSION command rules, a parameter from the clause_name parameter. To find existing parameters for this command rule, query the DBA_DV_COMMAND_RULE view.
event_name	For ALTER SYSTEM and ALTER SESSION command rules, an event that the command rule defines To find existing event names for this command rule, query the DBA_DV_COMMAND_RULE view.
component_name	A component of the event_name setting To find existing component names for this command rule, query the DBA_DV_COMMAND_RULE view.
action_name	An action of the component_name setting. To find existing action names for this command rule, query the DBA_DV_COMMAND_RULE view.
scope	Determines how to execute this procedure. The default is local. Options are as follows: • DBMS_MACUTL.G_SCOPE_LOCAL (or 1) if the command rule is local in the current PDB • DBMS_MACUTL.G_SCOPE_COMMON (or 2) if the command rule is in the application root

Example

The following example shows how to delete a common command rule from a Database Vault policy. This command rule is in the application root of a multitenant environment, so the user running this procedure must be in the CDB root.

```
BEGIN
 DBMS_MACADM.DELETE_CMD_RULE_FROM_POLICY(
  policy_name    => 'HR_DV_Policy',
  command        => 'ALTER SESSION',
  object_owner   => '%',
  object_name    => '%',
  clause_name    => 'END SESSION',
  parameter_name => 'KILL SESSION',
  event_name     => '',
  action_name    => '',
  scope          => DBMS_MACUTL.G_SCOPE_COMMON);
END;
/
```

22.6 DELETE_OWNER_FROM_POLICY Procedure

The `DELETE_OWNER_FROM_POLICY` procedure enables you to remove an owner from an Oracle Database Vault policy.

You can remove owners from policies any time, regardless of the state (enabled or disabled) of the policy. The change takes effect immediately.

Syntax

```
DBMS_MACADM.DELETE_OWNER_FROM_POLICY(
  policy_name    IN VARCHAR2,
  owner_name     IN VARCHAR2);
```

Parameters

Table 22-6 DELETE_OWNER_FROM_POLICY Parameters

Parameter	Description
`policy_name`	Policy name. To find existing Database Vault policies in the current database instance, query the `DBA_DV_POLICY` view.
`owner_name`	User name. To find existing policy owners in the current instance, query the `DBA_DV_POLICY_OWNER` view.

Example

```
BEGIN
 DBMS_MACADM.DELETE_OWNER_FROM_POLICY(
  policy_name    => 'HR_DV_Policy',
  owner_name     => 'PSMITH');
END;
/
```

22.7 DELETE_REALM_FROM_POLICY Procedure

The `DELETE_REALM_FROM_POLICY` procedure enables you to remove an existing realm from an Oracle Database Vault policy.

You can remove realms from policies any time, regardless of the state (enabled or disabled) of the policy. The change takes effect immediately.

Syntax

```
DBMS_MACADM.DELETE_REALM_FROM_POLICY(
  policy_name    IN VARCHAR2,
  realm_name     IN VARCHAR2);
```

Parameters

Table 22-7 DELETE_REALM_FROM_POLICY Parameters

Parameter	Description
`policy_name`	Policy name. To find existing Database Vault policies in the current database instance, query the `DBA_DV_POLICY` view.
`realm_name`	Realm name. To find existing Database Vault realms in the current database instance, query the `DV_REALM` view.

Example

```
BEGIN
 DBMS_MACADM.DELETE_REALM_FROM_POLICY(
  policy_name     => 'HR_DV_Policy',
  realm_name      => 'HR Realm');
END;
/
```

22.8 DROP_POLICY Procedure

The `DROP_POLICY` procedure enables you to drop an existing Oracle Database Vault policy.

You can remove a policy at any time, regardless of the state (enabled or disabled) of the policy.

Syntax

```
DBMS_MACADM.DROP_POLICY(
  policy_name       IN VARCHAR2);
```

Parameters

Table 22-8 DROP_POLICY Parameters

Parameter	Description
`policy_name`	Policy name. To find existing Database Vault policies in the current database instance, query the `DBA_DV_POLICY` view.

Example

```
EXEC DBMS_MACADM.DROP_POLICY ('HR_DV_Policy');
```

22.9 RENAME_POLICY Procedure

The `UPDATE_POLICY_DESCRIPTION` procedure enables you to rename an existing Oracle Database Vault policy.

You can rename a policy at any time, regardless of the state (enabled or disabled) of the policy. The change takes effect immediately.

Syntax

```
DBMS_MACADM.RENAME_POLICY(
  policy_name      IN VARCHAR2,
  new_policy_name  IN VARCHAR2);
```

Parameters

Table 22-9 RENAME_POLICY Parameters

Parameter	Description
`policy_name`	Policy name. To find existing Database Vault policies in the current database instance, query the `DBA_DV_POLICY` view.
`new_policy_name`	New policy name, up to 128 characters in mixed case

Example

```
BEGIN
 DBMS_MACADM.RENAME_POLICY(
  policy_name      => 'HR_DV_Policy',
  new_policy_name  => 'HR_WEST_COAST_DV_Policy');
END;
/
```

22.10 UPDATE_POLICY_DESCRIPTION Procedure

The `UPDATE_POLICY_DESCRIPTION` procedure enables you to update the `description` field in an Oracle Database Vault policy.

Syntax

```
DBMS_MACADM.UPDATE_POLICY_DESCRIPTION(
  policy_name  IN VARCHAR2,
  description  IN VARCHAR2 DEFAULT);
```

Parameters

Table 22-10 UPDATE_POLICY_DESCRIPTION Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view.
description	New description of the purpose of the policy, up to 4000 characters in mixed-case

Example

```
BEGIN
 DBMS_MACADM.UPDATE_POLICY_DESCRIPTION(
  policy_name  => 'HR_DV_Policy',
  description  => 'HR schema protection policy');
END;
/
```

22.11 UPDATE_POLICY_STATE Procedure

The UPDATE_POLICY_STATE procedure enables you to update the policy_state field in an Oracle Database Vault policy.

Syntax

```
DBMS_MACADM.UPDATE_POLICY_STATE(
  policy_name    IN VARCHAR2,
  policy_state   IN NUMBER,
 pl_sql_stack  IN BOOLEAN DEFAULT);
```

Parameters

Table 22-11 UPDATE_POLICY_STATE Parameters

Parameter	Description
policy_name	Policy name. To find existing Database Vault policies in the current database instance, query the DBA_DV_POLICY view.
policy_state	Specifies how the policy is enabled. Possible values are: • DBMS_MACADM.G_ENABLED (1), which enables the policy after you create it • DBMS_MACADM.G_DISABLED (0), which disables the policy after you create it • DBMS_MACADM.G_SIMULATION (2), which sets the policy to simulation mode. In simulation mode, any violations to realms or command rules used in the policy are logged in a designated log table with sufficient information to describe the error, such as the user name or SQL statement used. See also Related Topics. • DBMS_MACADM.G_PARTIAL (3), which sets the policy to partial mode. In partial mode, the enforcement state of realms or command rules associated with the policy can be changed individually. See About Simulation Mode for more information about simulation mode

Table 22-11 (Cont.) UPDATE_POLICY_STATE Parameters

Parameter	Description
pl_sql_stack	When simulation mode is enabled, specifies whether to record the PL/SQL stack for failed operations. Enter TRUE to record the PL/SQL stack, FALSE to not record.

Example

```
BEGIN
 DBMS_MACADM.UPDATE_POLICY_STATE(
  policy_name   => 'HR_DV_Policy',
  policy_state  => DBMS_MACADM.G_DISABLED,
  pl_sql_stack  => TRUE);
END;
/
```

Related Topics

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

23
Oracle Database Vault API Reference

Oracle Database Vault provides a rich set of APIs, both in PL/SQL packages and in standalone procedures.

- DBMS_MACADM PL/SQL Package Contents
  The `DBMS_MACADM` package enables you to configure the realms, factors, rule sets, command rules, secure application roles, and Oracle Label Security policies.
- DBMS_MACSEC_ROLES PL/SQL Package Contents
  The `DBMS_MACSEC_ROLES` package enables you to check and set Oracle Database Vault secure application roles.
- DBMS_MACUTL PL/SQL Package Contents
  The `DBMS_MACUTL` PL/SQL package defines constants and utility methods that are commonly used by other Oracle Database Vault packages, such as error handling.
- CONFIGURE_DV PL/SQL Procedure
  The `CONFIGURE_DV` configures the initial two Oracle Database user accounts, which are granted the `DV_OWNER` and `DV_ACCTMGR` roles, respectively.
- DVF PL/SQL Interface Contents
  The `DVF` schema provides a set of factor-related PL/SQL functions.

23.1 DBMS_MACADM PL/SQL Package Contents

The `DBMS_MACADM` package enables you to configure the realms, factors, rule sets, command rules, secure application roles, and Oracle Label Security policies.

The `DBMS_MACADM` package is available only for users who have been granted the `DV_ADMIN` or `DV_OWNER` role.

DBMS_MACADM Realm Procedures

Table 23-1 lists the realm procedures in the `DBMS_MACADM` package.

Table 23-1 DBMS_MACADM Realm Procedures

Procedure	Description
`ADD_AUTH_TO_REALM` procedure	Authorizes a user or role to access a realm as an owner or a participant
`ADD_OBJECT_TO_REALM` procedure	Registers a set of objects for realm protection
`CREATE_REALM` procedure	Creates a realm
`DELETE_AUTH_FROM_REALM` procedure	Removes the authorization of a user or role to access a realm
`DELETE_OBJECT_FROM_REALM` procedure	Removes a set of objects from realm protection

Table 23-1 (Cont.) DBMS_MACADM Realm Procedures

Procedure	Description
DELETE_REALM procedure	Deletes a realm, including its related Database Vault configuration information that specifies who is authorized and what objects are protected
DELETE_REALM_CASCADE procedure	Deletes a realm, including its related Database Vault configuration information that specifies who is authorized and what objects are protected
RENAME_REALM procedure	Renames a realm. The name change takes effect everywhere the realm is used.
UPDATE_REALM procedure	Updates a realm
UPDATE_REALM_AUTH procedure	Updates the authorization of a user or role to access a realm

DBMS_MACADM Rule Set and Rule Procedures

Table 23-2 lists the rule set and rule procedures in the DBMS_MACADM package.

Table 23-2 DBMS_MACADM Rule Set and Rule Procedures

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

DBMS_MACADM Command Rule Procedures

Table 23-3 lists the command rule procedures in the DBMS_MACADM package.

Table 23-3 DBMS_MACADM Command Rule Procedures

Procedure	Description
CREATE_COMMAND_RULE procedure	Creates a command rule, associates it with a rule set, and lets you enable the command rule for rule checking with a rule set

Table 23-3 (Cont.) DBMS_MACADM Command Rule Procedures

Procedure	Description
CREATE_CONNECT_COMMAND_RULE procedure	Creates a CONNECT command rule
CREATE_SESSION_EVENT_CMD_RULE procedure	Creates a session event command rule, using the ALTER SESSION SQL statement
CREATE_SYSTEM_EVENT_CMD_RULE procedure	Creates a system event command rule, using the ALTER SYSTEM SQL statement
DELETE_COMMAND_RULE procedure	Drops a command rule declaration
DELETE_CONNECT_COMMAND_RULE procedure	Drops a CONNECT command rule declaration
DELETE_SESSION_EVENT_CMD_RULE procedure	Drops a SESSION_EVENT_CMD command rule declaration
DELETE_SYSTEM_EVENT_CMD_RULE procedure	Drops a SYSTEM_EVENT_CMD command rule declaration
UPDATE_COMMAND_RULE procedure	Updates a command rule declaration
UPDATE_CONNECT_COMMAND_RULE procedure	Updates a CONNECT command rule declaration
UPDATE_SESSION_EVENT_CMD_RULE procedure	Updates a SESSION_EVENT_CMD command rule declaration
UPDATE_SYSTEM_EVENT_CMD_RULE procedure	Updates a SYSTEM_EVENT_CMD command rule declaration

DBMS_MACADM Factor Procedures and Functions

lists the factor procedures and functions in the DBMS_MACADM package.

Table 23-4 DBMS_MACADM Factor Procedures and Functions

Procedure or Function	Description
ADD_FACTOR_LINK procedure	Specifies a parent-child relationship for two factors
ADD_POLICY_FACTOR procedure	Specifies that the label for a factor contributes to the Oracle Label Security label for a policy.
CHANGE_IDENTITY_FACTOR procedure	Associates an identity with a different factor
CHANGE_IDENTITY_VALUE procedure	Updates the value of an identity
CREATE_DOMAIN_IDENTITY procedure	Adds an Oracle Real Application Clusters (Oracle RAC) database node to the domain factor identities and labels it according to the Oracle Label Security policy.
CREATE_FACTOR procedure	Creates a factor
CREATE_FACTOR_TYPE procedure	Creates a factor type
CREATE_IDENTITY procedure	Creates an identity
CREATE_IDENTITY_MAP procedure	Defines a set of tests that are used to derive the identity of a factor from the value of linked child factors (subfactors)
DELETE_FACTOR procedure	Deletes a factor

Table 23-4 (Cont.) DBMS_MACADM Factor Procedures and Functions

Procedure or Function	Description
DELETE_FACTOR_LINK procedure	Removes a parent-child relationship for two factors
DELETE_FACTOR_TYPE procedure	Deletes a factor type
DELETE_IDENTITY procedure	Removes an identity
DELETE_IDENTITY_MAP procedure	Removes an identity map from a factor
DROP_DOMAIN_IDENTITY procedure	Removes an Oracle RAC database node from a domain
GET_INSTANCE_INFO function	Returns information from the SYS.V_$INSTANCE system table about the current database instance; returns a VARCHAR2 value
GET_SESSION_INFO function	Returns information from the SYS.V_$SESSION system table for the current session; returns a VARCHAR2 value
RENAME_FACTOR procedure	Renames a factor. The name change takes effect everywhere the factor is used.
RENAME_FACTOR_TYPE procedure	Renames a factor type. The name change takes effect everywhere the factor type is used.
UPDATE_FACTOR procedure	Updates a factor
UPDATE_FACTOR_TYPE procedure	Updates the description of a factor type
UPDATE_IDENTITY procedure	Updates the trust level of a factor identity

DBMS_MACADM Secure Application Role Procedures

Table 23-5 lists the secure application role procedures in the DBMS_MACADM package.

Table 23-5 DBMS_MACADM Secure Application Role Procedures

Procedure	Description
CREATE_ROLE procedure	Creates an Oracle Database Vault secure application role
DELETE_ROLE procedure	Deletes an Oracle Database Vault secure application role
RENAME_ROLE procedure	Renames an Oracle Database Vault secure application role. The name change takes effect everywhere the role is used.
UNASSIGN_ROLE procedure	Unassigns an Oracle Database Vault secure application role from a user
UPDATE_ROLE procedure	Updates a Oracle Database Vault secure application role

DBMS_MACADM Oracle Label Security Procedures

Table 23-6 lists the Oracle Label Security procedures in the DBMS_MACADM package.

Table 23-6 DBMS_MACADM Oracle Label Security Procedures

Procedure	Description
CREATE_MAC_POLICY procedure	Specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label
CREATE_POLICY_LABEL procedure	Labels an identity within an Oracle Label Security policy
DELETE_MAC_POLICY_CASCADE procedure	Deletes all Oracle Database Vault objects related to an Oracle Label Security policy.
DELETE_POLICY_FACTOR procedure	Removes the factor from contributing to the Oracle Label Security label
DELETE_POLICY_LABEL procedure	Removes the label from an identity within an Oracle Label Security policy
UPDATE_MAC_POLICY procedure	Specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label

DBMS_MACADM Database Vault Policy Procedures

Table 23-7 lists the Database Vault policy procedures in the DBMS_MACADM package.

Table 23-7 DBMS_MACADM Database Vault Policy Procedures

Procedure	Description
ADD_CMD_RULE_TO_POLICY procedure	Adds a command rule to a Database Vault policy
ADD_OWNER_TO_POLICY procedure	Adds an owner to a Database Vault policy
ADD_REALM_TO_POLICY procedure	Adds a realm to a Database Vault policy
CREATE_POLICY procedure	Creates a Database Vault policy
DELETE_CMD_RULE_FROM_POLICY procedure	Deletes a command rule from a Database Vault policy
DELETE_OWNER_FROM_POLICY procedure	Deletes an owner from a Database Vault policy
DELETE_REALM_FROM_POLICY procedure	Deletes a realm from a Database Vault policy
DROP_POLICY procedure	Drops a Database Vault policy
RENAME_POLICY procedure	Renames a Database Vault policy
UPDATE_POLICY_DESCRIPTION procedure	Updates a Database Vault policy description
UPDATE_POLICY_STATE procedure	Updates the enablement status of the a Database Vault policy

DBMS_MACADM General Administrative Procedures

Table 23-8 lists the general administrative procedures in the DBMS_MACADM package.

Table 23-8 DBMS_MACADM General Administrative Procedures

Procedure	Description
ADD_NLS_DATA procedure	Adds a new language to Oracle Database Vault
ADD_APP_EXCEPTION procedure	Enables a common user or package to access local schemas
AUTHORIZE_DATAPUMP_USER procedure	Authorizes a user to perform Oracle Data Pump operations when Oracle Database Vault is enabled
AUTHORIZE_DDL procedure	Grants a user authorization to run data definition language (DDL) statements
AUTHORIZE_MAINTENANCE_USER procedure	Grants a user authorization to perform Information Lifecycle Management (ILM) operations
AUTHORIZE_PROXY_USER procedure	Grants a proxy user authorization to proxy other user accounts
AUTHORIZE_SCHEDULER_USER procedure	Authorizes a user to schedule database jobs when Oracle Database Vault is enabled
AUTHORIZE_TTS_USER procedure	Authorizes a user to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled
DELETE_APP_EXCEPTION procedure	Deletes the exception for a common user or package to access a local schema
DISABLE_DV_DICTIONARY_ACCTS procedure	Prevents users from logging into the DVSYS and DFV schema accounts
DISABLE_DV_PATCH_ADMIN	Disables auditing of the DV_PATCH_ADMIN user
DISABLE_DV procedure	Disables Oracle Database Vault
DISABLE_APP_PROTECTION procedure	Disables Database Vault operation control
DISABLE_ORADEBUG procedure	Disables the use of the ORADEBUG utility in an Oracle Database Vault environment
ENABLE_DV_DICTIONARY_ACCTS procedure	Enables users to log into the DVSYS and DFV schema accounts
ENABLE_DV_PATCH_ADMIN	Enables auditing of the DV_PATCH_ADMIN user
ENABLE_DV procedure	Enables Oracle Database Vault
ENABLE_APP_PROTECTION procedure	Enables Database Vault operations control
ENABLE_ORADEBUG procedure	Enables the use of the ORADEBUG utility in an Oracle Database Vault environment
UNAUTHORIZE_DATAPUMP_USER procedure	Revokes the authorization that was granted by the DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure
UNAUTHORIZE_DDL procedure	Revokes authorization from a user who was granted authorization to run DDL statements through the DBMS_MACDM.AUTHORIZE_DDL procedure
UNAUTHORIZE_MAINTENANCE_USER procedure	Revokes authorization to perform ILM operations
UNAUTHORIZE_PROXY_USER procedure	Revokes authorization from a user who was granted proxy authorization from the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure

Table 23-8 (Cont.) DBMS_MACADM General Administrative Procedures

Procedure	Description
UNAUTHORIZE_SCHEDULER_USER procedure	Revokes authorization that was granted by the DBMS_MACADM.AUTHORIZE_SCHEDULER_USER procedure
UNAUTHORIZE_TTS_USER procedure	Revokes from authorization a user who had been granted authorization to perform Oracle Data Pump transportable tablespace operations for a tablespace when Oracle Database Vault is enabled

23.2 DBMS_MACSEC_ROLES PL/SQL Package Contents

The DBMS_MACSEC_ROLES package enables you to check and set Oracle Database Vault secure application roles.

This package is available to the general database account population.

Table 23-9 lists the contents of the DBMS_MACSEC_ROLES package.

Table 23-9 DBMS_MACSEC_ROLES PL/SQL Package Contents

Procedure or Function	Description
CAN_SET_ROLE function	Checks whether the user invoking the method is authorized to use the specified Oracle Database Vault secure application role. Returns a BOOLEAN value.
SET_ROLE procedure	Issues the SET ROLE statement for an Oracle Database Vault secure application role.

23.3 DBMS_MACUTL PL/SQL Package Contents

The DBMS_MACUTL PL/SQL package defines constants and utility methods that are commonly used by other Oracle Database Vault packages, such as error handling.

This package can be run by the general database account population. This allows for security developers to leverage the constants in scripted configuration files. Utility methods such as USER_HAS_ROLE can also be used in Oracle Database Vault rules.

Table 23-10 lists the DBMS_MACUTL package contents.

Table 23-10 DBMS_MACUTL PL/SQL Package Contents

Procedure or Function	Description
CHECK_DVSYS_DML_ALLOWED procedure	Verifies that public-packages are not being bypassed by users updating the Oracle Database Vault configuration
GET_CODE_VALUE function	Looks up the value for a code within a code group.
GET_SECOND function	Returns the seconds in Oracle SS format (00-59). Useful for rule expressions based on time data

Table 23-10 (Cont.) DBMS_MACUTL PL/SQL Package Contents

Procedure or Function	Description
GET_MINUTE function	Returns the minute in Oracle MI format (00–59). Useful for rule expressions based on time data
GET_HOUR function	Returns the month in Oracle HH24 format (00–23). Useful for rule expressions based on time data
GET_DAY function	Returns the day in Oracle DD format (01–31). Useful for rule expressions based on time data
GET_MONTH function	Returns the month in Oracle MM format (01–12). Useful for rule expressions based on time data
GET_YEAR function	Returns the year in Oracle YYYY format (0001–9999). Useful for rule expressions based on time data
IS_ALPHA function	Checks whether the character is alphabetic
IS_DIGIT function	Checks whether the character is numeric
IS_DVSYS_OWNER function	Determines whether a user is authorized to manage the Oracle Database Vault configuration
IS_OLS_INSTALLED function	Returns an indicator regarding whether Oracle Label Security is installed
IS_OLS_INSTALLED_VARCHAR function	Returns an indicator regarding whether Oracle Label Security is installed
USER_HAS_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)

23.4 CONFIGURE_DV PL/SQL Procedure

The CONFIGURE_DV configures the initial two Oracle Database user accounts, which are granted the DV_OWNER and DV_ACCTMGR roles, respectively.

This procedure is used as part of the registration process for Oracle Database Vault with an Oracle database. You only need to use it once for the database instance.

23.5 DVF PL/SQL Interface Contents

The DVF schema provides a set of factor-related PL/SQL functions.

The functions are then available to the general database account population through PL/SQL functions and standard SQL.

Table 23-11 lists the DVF factor functions.

Table 23-11 DVF PL/SQL Interface Contents

Function	Description
F$CLIENT_IP	Returns the IP address of the computer from which the client is connected
F$DATABASE_DOMAIN	Returns the domain of the database as specified in the DB_DOMAIN initialization parameter
F$DATABASE_HOSTNAME	Returns the host name of the computer on which the database instance is running
F$DATABASE_INSTANCE	Returns the database instance identification number of the current database instance
F$DATABASE_IP	Returns the IP address of the computer on which the database instance is running
F$DATABASE_NAME	Returns the name of the database as specified in the DB_NAME initialization parameter
F$DOMAIN	Returns a named collection of physical, configuration, or implementation-specific factors in the run-time environment (for example, a networked IT environment or subset of it) that operates at a specific sensitivity level
F$ENTERPRISE_IDENTITY	Returns the enterprise-wide identity for a user
F$IDENTIFICATION_TYPE	Returns the way the schema of a user was created in the database. Specifically, it reflects the IDENTIFIED clause in the CREATE USER or ALTER USER syntax.
F$LANG	Returns the ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter
F$LANGUAGE	Returns the language and territory currently used by your session, in VARCHAR2 data type, along with the database character set
F$MACHINE	Returns the computer (host) name for the database client that established the database session.
F$NETWORK_PROTOCOL	Returns the network protocol being used for communication, as specified in the PROTOCOL=*protocol* portion of the connect string
F$PROXY_ENTERPRISE_IDENTITY	Returns the Oracle Internet Directory distinguished name (DN) when the proxy user is an enterprise user
F$SESSION_USER	Returns the database user name by which the current user is authenticated

24

Oracle Database Vault Data Dictionary Views

You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

- About the Oracle Database Vault Data Dictionary Views
  You can use both the traditional auditing data dictionary views that Oracle Database Vault provides, as well as unified auditing views to find information about Oracle Database Vault policies and other settings.
- CDB_DV_STATUS View
  The `CDB_DV_STATUS` data dictionary view shows the Database Vault operations control, configuration, and enablement status for all PDBs.
- DBA_DV_APP_EXCEPTION View
  The `DBA_DV_APP_EXCEPTION` data dictionary view lists the common schemas and package names that are in the Database Vault operations control exception list.
- DBA_DV_AUDIT_ADMIN_AUTH View
  The `DBA_DV_AUDIT_ADMIN_AUTH` data dictionary view lists users who have `AUDIT_ADMIN` authorization.
- DBA_DV_AUDIT_VIEWER_AUTH View
  The `DBA_DV_AUDIT_VIEWER_AUTH` data dictionary view lists users who have `AUDIT_VIEWER` authorization.
- DBA_DV_CODE View
  The `DBA_DV_CODE` data dictionary view lists generic lookup codes for the user interface, error messages, and constraint checking.
- DBA_DV_COMMAND_RULE View
  The `DBA_DV_COMMAND_RULE` data dictionary view lists the SQL statements that are protected by command rules.
- DBA_DV_DATAPUMP_AUTH View
  The `DBA_DV_DATAPUMP_AUTH` data dictionary view lists the authorizations for using Oracle Data Pump in an Oracle Database Vault environment.
- DBA_DV_DBCAPTURE_AUTH View
  The `DBA_DV_DBCAPTURE_AUTH` data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload capture operations.
- DBA_DV_DBREPLAY View
  The `DBA_DV_DBREPLAY_AUTH` data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload replay operations.
- DBA_DV_DDL_AUTH View
  The `DBA_DV_DDL` data dictionary view lists the users and schemas that were specified by the `DBMS_MACADM.AUTHORIZE_DDL` procedure.
- DBA_DV_DICTIONARY_ACCTS View
  The `DBA_DV_DICTIONARY_ACCTS` data dictionary view indicates whether users can directly log into the `DVSYS` and `DVF` schema accounts.

- DBA_DV_FACTOR View
  The DBA_DV_FACTOR data dictionary view lists the existing factors in the current PDB.
- 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.
- DBA_DV_FACTOR_LINK View
  The DBA_DV_FACTOR_LINK data dictionary view shows the relationships of each factor whose identity is determined by the association of child factors.
- DBA_DV_IDENTITY View
  The DBA_DV_IDENTITY data dictionary view lists the identities for each factor.
- DBA_DV_IDENTITY_MAP View
  The DBA_DV_IDENTITY_MAP data dictionary view lists the mappings for each factor identity.
- DBA_DV_JOB_AUTH View
  The DBA_DV_JOB_AUTH data dictionary view lists the authorizations for using Oracle Scheduler in an Oracle Database Vault environment.
- DBA_DV_MAC_POLICY View
  The DBA_DV_MAC_POLICY data dictionary view lists the Oracle Label Security policies defined for use with Oracle Database Vault.
- DBA_DV_MAC_POLICY_FACTOR View
  The DBA_DV_MAC_POLICY data dictionary view lists the factors that are associated with Oracle Label Security policies.
- DBA_DV_MAINTENANCE_AUTH View
  The DBA_DV_MAINTENANCE_AUTH data dictionary view provides information about the configuration of Oracle Database Vault authorizations to use Information Life Management (ILM) features.
- DBA_DV_ORADEBUG View
  The DBA_DV_ORADEBUG data dictionary view indicates whether users can use the ORADEBUG utility in an Oracle Database Vault environment.
- DBA_DV_PATCH_ADMIN_AUDIT View
  The DBA_DV_PATCH_ADMIN_AUDIT data dictionary view indicates if auditing has been enabled or disabled for the user who has been granted the DV_ADMIN_PATCH role.
- DBA_DV_POLICY View
  The DBA_DV_POLICY data dictionary view lists the Oracle Database Vault policies that were created in the current database instance.
- DBA_DV_POLICY_LABEL View
  The DBA_DV_POLICY_LABEL data dictionary view lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.
- DBA_DV_POLICY_OBJECT View
  The DBA_DV_POLICY_OBJECT data dictionary view lists information about the objects that are protected by Oracle Database Vault policies in the current database instance.
- DBA_DV_POLICY_OWNER View
  The DBA_DV_POLICY_OWNER data dictionary view lists the owners of Oracle Database Vault policies that were created in the current database instance.

- DBA_DV_PREPROCESSOR_AUTH View
  The DBA_DV_PREPROCESSOR_AUTH data dictionary view shows users who have been granted authorization to run preprocessor programs through external tables.
- DBA_DV_PROXY_AUTH View
  The DBA_DV_PROXY_AUTH data dictionary view lists the proxy users and schemas that were specified by the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure.
- DBA_DV_PUB_PRIVS View
  The DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in Oracle Database Vault Administrator.
- DBA_DV_REALM View
  The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.
- DBA_DV_REALM_AUTH View
  The DBA_DV_REALM_AUTH data dictionary view lists database user account or role authorization (GRANTEE) who can access realm objects.
- DBA_DV_REALM_OBJECT View
  The DBA_DV_REALM_OBJECT data dictionary view lists the database schemas, or subsets of schemas, that are secured by the realms.
- DBA_DV_ROLE View
  The DBA_DV_ROLE data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.
- DBA_DV_RULE View
  The DBA_DV_RULE data dictionary view lists the rules that have been defined.
- DBA_DV_RULE_SET View
  The DBA_DV_RULE_SET data dictionary view lists the rules sets that have been created.
- DBA_DV_RULE_SET_RULE View
  The DBA_DV_RULE_SET_RULE data dictionary view lists rules that are associated with existing rule sets.
- DBA_DV_SIMULATION_LOG View
  The DBA_DV_SIMULATION_LOG data dictionary view captures simulation log information for realms and command rules that have had simulation mode enabled.
- DBA_DV_STATUS or SYS.DBA_DV_STATUS View
  The DBA_DV_STATUS (or SYS.DBA_DV_STATUS) data dictionary view shows the status of Oracle Database Vault being enabled and configured.
- DBA_DV_SQL_FIREWALL_AUTH View
  The DBA_DV_SQL_FIREWALL_AUTH data dictionary view shows users who have been granted authorization to perform SQL Firewall operations.
- DBA_DV_TTS_AUTH View
  The DBA_DV_TTS_AUTH data dictionary view lists users who have been granted authorization through the DBMS_MACADM.AUTHORIZE_TTS_USER procedure to perform Oracle Data Pump transportable operations.
- DBA_DV_USER_PRIVS View
  The DBA_DV_USER_PRIVS data dictionary view lists the privileges for a database user account excluding privileges granted through the PUBLIC role.

- DBA_DV_USER_PRIVS_ALL View
  The DBA_DV_USER_PRIVS_ALL data dictionary view lists the privileges for a database account including privileges granted through PUBLIC.
- DVSYS.DV$CONFIGURATION_AUDIT View
  The DVSYS.DV$CONFIGURATION_AUDIT data dictionary view captures DVSYS.AUDIT_TRAIL$ table audit trail records.
- DVSYS.DV$ENFORCEMENT_AUDIT View
  The DVSYS.DV$ENFORCEMENT_AUDIT data dictionary view provides information about enforcement-related audits from the DVSYS.AUDIT_TRAIL$ table.
- DVSYS.DV$REALM View
  The DVSYS.DV$REALM data dictionary view describes settings that were used to create Oracle Database Vault realms, such as which audit options have been assigned or whether the realm is a mandatory realm.
- DVSYS.DBA_DV_COMMON_OPERATION_STATUS View
  The DVSYS.DBA_DV_COMMON_OPERATION_STATUS data dictionary view displays the status of the DBMS_MACADM.ALLOW_COMMON_OPERATION procedure setting.
- DVSYS.POLICY_OWNER_COMMAND_RULE View
  The DVSYS.POLICY_OWNER_COMMAND_RULE data dictionary view enables DV_POLICY_OWNER role users to find information about the command rules that are used by Database Vault policies.
- DVSYS.POLICY_OWNER_POLICY View
  The DVSYS.POLICY_OWNER_POLICY data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners.
- DVSYS.POLICY_OWNER_REALM View
  The POLICY_OWNER_REALM data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the realms that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_REALM_AUTH View
  The DVSYS.POLICY_OWNER_REALM_AUTH data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the authorization that was granted to realms that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_REALM_OBJECT View
  The DVSYS.POLICY_OWNER_REALM_OBJECT data dictionary view enables users to find information about the objects that have been added to realms that are associated with Database Vault policies, such as. Only users who have been granted the DV_POLICY_OWNER role can query this view.
- DVSYS.POLICY_OWNER_RULE View
  The DVSYS.POLICY_OWNER_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rules that have been associated with rule sets in Database Vault policies, such as the rule name and its expression. Only users who have been granted the DV_POLICY_OWNER role can query this view.

- DVSYS.POLICY_OWNER_RULE_SET View
  The DVSYS.POLICY_OWNER_RULE_SET data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_RULE_SET_RULE View
  The DVSYS.POLICY_OWNER_RULE_SET_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that contain rules used in Database Vault policies.
- AUDSYS.DV$CONFIGURATION_AUDIT View
  The AUDSYS.DV$CONFIGURATION_AUDIT view is almost the same as the DVSYS.DV$CONFIGURATION_AUDIT view except that it captures unified audit trail Database Vault audit records.
- AUDSYS.DV$ENFORCEMENT_AUDIT View
  The AUDSYS.DV$ENFORCEMENT_AUDIT view is almost the same as the DVSYS.DV$ENFORCEMENT_AUDIT view except that it captures unified audit trail Database Vault audit records.

24.1 About the Oracle Database Vault Data Dictionary Views

You can use both the traditional auditing data dictionary views that Oracle Database Vault provides, as well as unified auditing views to find information about Oracle Database Vault policies and other settings.

- **Traditional auditing:** The Oracle Database Vault DBA-style data dictionary views that can be accessed through the DV_SECANALYST role or the DV_ADMIN role are used to query audit records from traditional auditing settings.
- **Unified auditing:** In addition to views such as UNIFIED_AUDIT_TRAIL, users who have the AUDIT_ADMIN and AUDIT_VIEWER roles can query Oracle Database Vault-specific unified audit trail views.

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.

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.

24.2 CDB_DV_STATUS View

The CDB_DV_STATUS data dictionary view shows the Database Vault operations control, configuration, and enablement status for all PDBs.

Only Oracle Database administrative users, such users who have been granted the DBA role, can query this view. Database Vault administrators do not have access to this view.

For example:

```
SELECT * FROM CDB_DV_STATUS;
```

Output similar to the following appears:

```
NAME                 STATUS  CON_ID
-------------------- ------- ------
DV_APP_PROTECTION    ENABLED      5
DV_CONFIGURE_STATUS  TRUE         5
DV_ENABLE_STATUS     TRUE         5
```

Column	Datatype	Null	Description
NAME	VARCHAR2(19)	NOT NULL	Shows either of the following settings: • DV_APP_PROTECTION shows whether Database Vault operations control is enabled or not enabled. • DV_CONFIGURE_STATUS shows whether Oracle Database Vault is configured (that is, with the CONFIGURE_DV procedure). • DV_ENABLE_STATUS shows whether Oracle Database Vault is enabled (that is, with the DBMS_MACADM.ENABLE_DV procedure).
STATUS	VARCHAR2(64)	NOT NULL	For DV_CONFIGURE_STATUS and DV_ENABLE_STATUS, TRUE means that Oracle Database Vault is configured or enabled; FALSE means that it is not. For DV_APP_PROTECTION, the output is ENABLED or DISABLED.
CON_ID	NUMBER	NOT NULL	The identification number of the PDB container in which Oracle Database Vault is used

Related Topics

- DBA_DV_STATUS or SYS.DBA_DV_STATUS View
  The DBA_DV_STATUS (or SYS.DBA_DV_STATUS) data dictionary view shows the status of Oracle Database Vault being enabled and configured.

24.3 DBA_DV_APP_EXCEPTION View

The DBA_DV_APP_EXCEPTION data dictionary view lists the common schemas and package names that are in the Database Vault operations control exception list.

You must query this view from the CDB root only. If you try to query this view from a pluggable database (PDB), then no output appears.

For example:

```
SELECT * FROM DBA_DV_APP_EXCEPTION WHERE GRANTEE = 'C##HR_ADMIN';
```

Output similar to the following appears:

```
GRANTEE       PACKAGE_NAME
------------- ------------
C##HR_ADMIN     PATCH_APP
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR(128)	NOT NULL	Name of the grantee To find the names of common users, query the USERNAME and COMMON columns of the DBA_USERS data dictionary view.
PACKAGE_NAME	VARCHAR(128)	NOT NULL	Name of the package

24.4 DBA_DV_AUDIT_ADMIN_AUTH View

The DBA_DV_AUDIT_ADMIN_AUTH data dictionary view lists users who have AUDIT_ADMIN authorization.

For example:

```
SELECT * FROM DBA_DV_AUDIT_ADMIN_AUTH WHERE GRANTEE = 'PSMITH';
```

Output similar to the following appears:

```
GRANTEE
-----------
PSMITH
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted AUDIT_ADMIN authorization

Related Topics

- DBA_DV_AUDIT_VIEWER_AUTH View
  The DBA_DV_AUDIT_VIEWER_AUTH data dictionary view lists users who have AUDIT_VIEWER authorization.

24.5 DBA_DV_AUDIT_VIEWER_AUTH View

The DBA_DV_AUDIT_VIEWER_AUTH data dictionary view lists users who have AUDIT_VIEWER authorization.

For example:

```
SELECT * FROM DBA_DV_AUDIT_VIEWER_AUTH WHERE UNAME = 'RLAYTON';
```

Output similar to the following appears:

```
GRANTEE
-----------
RLAYTON
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted AUDIT_VIEWER authorization.

Related Topics

- DBA_DV_AUDIT_ADMIN_AUTH View
  The DBA_DV_AUDIT_ADMIN_AUTH data dictionary view lists users who have AUDIT_ADMIN authorization.

24.6 DBA_DV_CODE View

The DBA_DV_CODE data dictionary view lists generic lookup codes for the user interface, error messages, and constraint checking.

These codes are used for the user interface, views, and for validating input in a translatable fashion.

For example:

```
SELECT CODE, VALUE FROM DBA_DV_CODE WHERE CODE_GROUP = 'BOOLEAN';
```

Output similar to the following appears:

```
CODE    VALUE
------- --------
Y       True
N       False
```

Column	Datatype	Null	Description
CODE_GROUP	VARCHAR(128)	NOT NULL	Displays one of the code groups that are listed in Table 24-1
CODE	VARCHAR(128)	NOT NULL	Boolean code used; either Y (Yes) or N (No).
VALUE	VARCHAR(4000)	NULL	Boolean value used; either True if the Boolean code is Y or False if the Boolean code is N.
LANGUAGE	VARCHAR(3)	NOT NULL	Language for this installation of Oracle Database Vault. Supported languages are as follows: • en: English • de: German • es: Spanish • fr: French • it: Italian • ja: Japanese • ko: Korean • pt_BR: Brazilian Portuguese • zh_CN: Simplified Chinese • zh_TW: Traditional Chinese
DESCRIPTION	VARCHAR(1024)	NULL	Brief description of the code group.

Table 24-1 describes the possible values from the CODE_GROUP column in the DBA_DV_CODE data dictionary view.

Table 24-1 DBA_DV_CODE View CODE_GROUP Values

CODE_GROUP Name	Description
AUDIT_EVENTS	Contains the action numbers and action names that are used for the custom event audit trail records

Table 24-1 (Cont.) DBA_DV_CODE View CODE_GROUP Values

CODE_GROUP Name	Description
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 Related Topics.
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

Related Topics

- Table 19-2

24.7 DBA_DV_COMMAND_RULE View

The DBA_DV_COMMAND_RULE data dictionary view lists the SQL statements that are protected by command rules.

For example:

```
SELECT COMMAND, RULE_SET_NAME FROM DBA_DV_COMMAND_RULE;
```

Output similar to the following appears:

```
COMMAND         RULE_SET_NAME
--------------- ----------------------------
GRANT           Can Grant VPD Administration
```

```
REVOKE          Can Grant VPD Administration
ALTER SYSTEM    Allow System Parameters
ALTER USER      Can Maintain Own Account
CREATE USER     Can Maintain Account/Profiles
DROP USER       Can Maintain Account/Profiles
CREATE PROFILE  Can Maintain Account/Profiles
DROP PROFILE    Can Maintain Account/Profiles
ALTER PROFILE   Can Maintain Account/Profiles
```

Column	Datatype	Null	Description
COMMAND	VARCHAR(128)	NOT NULL	Name of the command rule.
CLAUSE_NAME	VARCHAR(100)	NOT NULL	A clause from either the ALTER SYSTEM or ALTER SESSION SQL statement, which was used to create the command rule. For example, you it could list the SET clause for the ALTER SESSION statement. The command rule settings for these two statements are described in the DBMS_MACADM.CREATE_COMMAND_RULE procedure. See Related Topics.
PARAMETER_NAME	VARCHAR(128)	NOT NULL	A parameter from the ALTER SYSTEM or ALTER SESSION command rule CLAUSE_NAME setting. See Related Topics.
EVENT_NAME	VARCHAR(128)	NOT NULL	An event that the ALTER SYSTEM or ALTER SESSION command rule defines. See Related Topics.
COMPONENT_NAME	VARCHAR(128)	NOT NULL	A component of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule. See Related Topics.
ACTION_NAME	VARCHAR(128)	NOT NULL	An action of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule. See Related Topics.
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set associated with this command rule.
OBJECT_OWNER	VARCHAR(128)	NOT NULL	The owner of the object that the command rule affects.
OBJECT_NAME	VARCHAR(128)	NOT NULL	The name of the database object the command rule affects (for example, a database table).
ENABLED	VARCHAR(1)	NOT NULL	Possible values are as follows: • Y indicates the command rule is enabled • N indicates it is disabled • S indicates it is in simulation mode
PRIVILEGE_SCOPE	NUMBER	NOT NULL	Obsolete column
COMMON	VARCHAR(3)	NOT NULL	Indicates whether the command rule is local or common. Possible values are: • YES if the command rule is common • NO if the command rule is local

Column	Datatype	Null	Description
INHERITED	VARCHAR(3)	NOT NULL	Shows the inheritance status of the command rule, when the COMMON column output is YES. Values are as follows: • YES means that the command rule was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the command rule is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ID#	NUMBER	NOT NULL	The ID number of the command rule, which is automatically generated when the command rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the command rule is a default (that is, Oracle-supplied) command rule or a user-created command rule. Possible values are: • YES if the command rule is a default command rule • NO if the command rule is a user-created command rule
PL_SQL_STACK	VARCHAR(3)	NULL	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

Related Topics

- Configuring Command Rules
  You can create command rules or use the default command rules to protect DDL and DML statements.
- CREATE_COMMAND_RULE Procedure
  The CREATE_COMMAND_RULE procedure creates both command and local command rules, which can be added to a rule set.

24.8 DBA_DV_DATAPUMP_AUTH View

The DBA_DV_DATAPUMP_AUTH data dictionary view lists the authorizations for using Oracle Data Pump in an Oracle Database Vault environment.

For example:

```
SELECT * FROM DBA_DV_DATAPUMP_AUTH WHERE GRANTEE = 'PRESTON';
```

Output similar to the following appears:

```
GRANTEE SCHEMA OBJECT  TYPE  ACTION
------- ------ ------- ----- -----------
PRESTON OE     ORDERS  %     CREATE_USER
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Data Pump authorization
SCHEMA	VARCHAR2(128)	NOT NULL	Name of the schema on which the user GRANTEE is authorized to perform Data Pump operations
OBJECT	VARCHAR2(128)	NOT NULL	Name of the object within the schema specified by the SCHEMA parameter on which the GRANTEE user has Data Pump authorization (such as a table)
TYPE	VARCHAR2(32)	NOT NULL	For Oracle Data Pump import operations, indicates the type of grant, such as ROLE)
ACTION	VARCHAR2(30)	NOT NULL	For Oracle Data Pump import operations, indicates action that is associated with the TYPE: %, TABLE, CREATE_USER, or GRANT

Related Topics

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

24.9 DBA_DV_DBCAPTURE_AUTH View

The DBA_DV_DBCAPTURE_AUTH data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload capture operations.

For example:

```
SELECT * FROM DBA_DV_DBCAPTURE_AUTH WHERE GRANTEE = 'PFITCH';
```

Output similar to the following appears:

```
GRANTEE
-------
PFITCH
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Database Replay workload capture authorization

Related Topics

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

24.10 DBA_DV_DBREPLAY View

The DBA_DV_DBREPLAY_AUTH data dictionary view shows users who have been granted authorization to perform Oracle Database Replay workload replay operations.

For example:

```
SELECT * FROM DBA_DV_DBREPLAY_AUTH WHERE GRANTEE = 'PFITCH';
```

Output similar to the following appears:

```
GRANTEE
-------
PFITCH
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Database Replay workload replay authorization

Related Topics

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

24.11 DBA_DV_DDL_AUTH View

The DBA_DV_DDL data dictionary view lists the users and schemas that were specified by the DBMS_MACADM.AUTHORIZE_DDL procedure.

This procedure grants a user authorization to run Data Definition Language (DDL) statements.

For example:

```
SELECT * FROM DBA_DV_DDL_AUTH WHERE GRANTEE = 'psmith';
```

Output similar to the following appears:

```
GRANTEE SCHEMA
------- ------
PSMITH  HR
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted DDL authorization
SCHEMA	VARCHAR2(128)	NOT NULL	Name of the schema on which the user GRANTEE is authorized to perform DDL operations

Related Topics

- AUTHORIZE_DDL Procedure
  The AUTHORIZE_DDL procedure grants a user authorization to run Data Definition Language (DDL) statements on the specified schema.

- UNAUTHORIZE_DDL Procedure
  The UNAUTHORIZE_DDL procedure revokes authorization from a user who was granted authorization to run DDL statements through the DBMS_MACDM.AUTHORIZE_DDL procedure.

24.12 DBA_DV_DICTIONARY_ACCTS View

The DBA_DV_DICTIONARY_ACCTS data dictionary view indicates whether users can directly log into the DVSYS and DVF schema accounts.

For example:

```
SELECT * FROM DBA_DV_DICTIONARY_ACCTS;
```

Output similar to the following appears:

```
STATE
-------
ENABLED
```

Column	Datatype	Null	Description
STATE	VARCHAR2(8)	NOT NULL	Describes whether users can log directly into the DVSYS and DVF schemas. Possible values are: • ENABLED means that users can log directly into the DVSYS and DVF schemas • DISABLED means that users cannot log directly into the DVSYS and DVF schemas

24.13 DBA_DV_FACTOR View

The DBA_DV_FACTOR data dictionary view lists the existing factors in the current PDB.

For example:

```
SELECT NAME, GET_EXPR FROM DBA_DV_FACTOR WHERE NAME = 'Session_User';
```

Output similar to the following appears:

```
NAME          GET_EXPR
------------- -------------------------------------------
Session_User  UPPER(SYS_CONTEXT('USERENV', 'SESSION_USER'))
```

Column	Datatype	Null	Description
NAME	VARCHAR2(128)	NOT NULL	Name of the factor.
DESCRIPTION	VARCHAR2(4000)	NULL	Description of the factor.
FACTOR_TYPE_NAME	VARCHAR2(128)	NOT NULL	Category of the factor, which is used to classify the purpose of the factor.
ASSIGN_RULE_SET_NAME	VARCHAR2(128)	NULL	Rule set used to control the identify of the factor.
GET_EXPR	VARCHAR2(1024)	NULL	PL/SQL expression that retrieves the identity of a factor.

Column	Datatype	Null	Description
VALIDATE_EXPR	VARCHAR2(1024)	NULL	PL/SQL expression used to validate the identify of the factor. It returns a Boolean value.
IDENTIFIED_BY	NUMBER	NOT NULL	Determines the identity of a factor, based on the expression listed in the GET_EXPR column. Possible values are: • 0: By constant • 1: By method • 2: By factors
IDENTIFIED_BY_MEANING	VARCHAR2(4000)	NULL	Provides a text description for the corresponding value in the IDENTIFIED_BY column. Possible values are: • By Constant: If IDENTIFIED_COLUMN is 0 • By Method: If IDENTIFIED_COLUMN is 1 • By Factors: If IDENTIFIED_COLUMN is 2
LABELED_BY	NUMBER	NOT NULL	Determines the labeling the factor: • 0: Labels the identities for the factor directly from the labels associated with an Oracle Label Security policy • 1: Derives the factor identity label from the labels of its child factor identities.
LABELED_BY_MEANING	VARCHAR2(4000)	NULL	Provides a text description for the corresponding value in the LABELED_BY column. Possible values are: • By Self: If LABELED_BY column is 0 • By Factors: If LABELED_BY column is 1
EVAL_OPTIONS	NUMBER	NOT NULL	Determines how the factor is evaluated when the user logs on: • 0: When the database session is created • 1: Each time the factor is accessed • 2: On start-up
EVAL_OPTIONS_MEANING	VARCHAR2(4000)	NULL	Provides a text description for the corresponding value in the EVAL_OPTIONS column. Possible values are: • For Session: If EVAL_OPTIONS is 0 • By Access: If EVAL_OPTIONS is 1 • On Startup: If EVAL_OPTIONS is 2

Column	Datatype	Null	Description
AUDIT_OPTIONS	NUMBER	NOT NULL	Option for auditing the factor using traditional auditing 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. Starting with Oracle Database 23c, traditional auditing is desupported.
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)	NULL	Provides a text description for the corresponding value in the FAIL_OPTIONS column. Possible values are: • Show Error Message • Do Not Show Error Message:
ID#	NUMBER	NOT NULL	The ID number of the factor, which is automatically generated when the factor is created
ORACLE_SUPPLIED	VARCHAR(3)	NOT NULL	Indicates whether the factor is a default (that is, Oracle-supplied) factor or a user-created factor. Possible values are: • YES if the factor is a default factor • NO if the factor is a user-created factor

Related Topics

- DBA_DV_FACTOR_LINK View
  The DBA_DV_FACTOR_LINK data dictionary view shows the relationships of each factor whose identity is determined by the association of child factors.
- DBA_DV_FACTOR_TYPE View
  The DBA_DV_FACTOR_TYPE data dictionary view lists the names and descriptions of factor types used in the system.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

24.14 DBA_DV_FACTOR_TYPE View

The DBA_DV_FACTOR_TYPE data dictionary view lists the names and descriptions of factor types used in the system.

For example:

```
SELECT * FROM DBA_DV_FACTOR_TYPE WHERE NAME = 'Time';
```

Output similar to the following appears:

```
NAME      DESCRIPTION
--------- --------------------------------------------------------------------
Time      Time-based factor
```

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Name of the factor type.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the factor type.

Related Topics

- DBA_DV_FACTOR View
  The DBA_DV_FACTOR data dictionary view lists the existing factors in the current PDB.
- 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.

24.15 DBA_DV_FACTOR_LINK View

The DBA_DV_FACTOR_LINK data dictionary view shows the relationships of each factor whose identity is determined by the association of child factors.

This view contains one entry for each parent factor and child factor. You can use this view to resolve the relationships from the factor links to identity maps.

For example:

```
SELECT PARENT_FACTOR_NAME, CHILD_FACTOR_NAME FROM DBA_DV_FACTOR_LINK;
```

Output similar to the following appears:

```
PARENT_FACTOR_NAME             CHILD_FACTOR_NAME
------------------------------ ------------------------------
Domain                         Database_Instance
Domain                         Database_IP
Domain                         Database_Hostname
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_FACTOR_TYPE View

Column	Datatype	Null	Description
PARENT_FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the parent factor
CHILD_FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the child factor of the parent factor
LABEL_IND	VARCHAR(1)	NOT NULL	Indicates whether the child factor that is linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Possible values are: • Y (for Yes) • N (for No)

Related Topics

- DBA_DV_FACTOR View
  The DBA_DV_FACTOR data dictionary view lists the existing factors in the current PDB.
- 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.

24.16 DBA_DV_IDENTITY View

The DBA_DV_IDENTITY data dictionary view lists the identities for each factor.

For example:

```
SELECT * FROM DBA_DV_IDENTITY WHERE VALUE = 'GLOBAL SHARED';
```

Output similar to the following appears, assuming you have created only one factor identity:

```
FACTOR_NAME          VALUE          TRUST_LEVEL
----------------     -------------- ------------
Identification_Type  GLOBAL SHARED  1
```

Related Views

- DBA_DV_FACTOR View
- DBA_DV_IDENTITY_MAP View

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the factor.
VALUE	VARCHAR(1024)	NOT NULL	Value of the factor.
TRUST_LEVEL	NUMBER	NOT NULL	Number that indicates the magnitude of trust relative to other identities for the same factor.

Related Topics

- DBA_DV_FACTOR View
  The DBA_DV_FACTOR data dictionary view lists the existing factors in the current PDB.

- DBA_DV_IDENTITY_MAP View
  The DBA_DV_IDENTITY_MAP data dictionary view lists the mappings for each factor identity.

24.17 DBA_DV_IDENTITY_MAP View

The DBA_DV_IDENTITY_MAP data dictionary view lists the mappings for each factor identity.

The view includes mapping factors that are identified by other factors to combinations of parent-child factor links. For each factor, the maps are joined by the OR operation, and for different factors, the maps are joined by the AND operation.

You can use this view to resolve the identity for factors that are identified by other factors (for example, a domain) or for factors that have continuous domains (for example, Age or Temperature).

For example:

```
SELECT FACTOR_NAME, IDENTITY_VALUE FROM DBA_DV_IDENTITY_MAP;
```

Output similar to the following appears:

```
FACTOR_NAME      IDENTITY_VALUE
---------------- --------------------
Sector2_Program  Accounting-Sensitive
```

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(128)	NOT NULL	Factor the identity map is for.
IDENTITY_VALUE	VARCHAR(1024)	NOT NULL	Value the factor assumes if the identity map evaluates to TRUE.
OPERATION_CODE	VARCHAR(128)	NOT NULL	Descriptive name of the operation in the OPERATION_VALUE column.
OPERATION_VALUE	VARCHAR(4000)	NULL	Relational operator for the identity map (for example, <, >, =, and so on).
OPERAND1	VARCHAR(1024)	NULL	Left operand for the relational operator; refers to the low value you enter.
OPERAND2	VARCHAR(1024)	NULL	Right operand for the relational operator; refers to the high value you enter.
PARENT_FACTOR_NAME	VARCHAR(128)	NULL	The parent factor link to which the map is related.
CHILD_FACTOR_NAME	VARCHAR(128)	NULL	The child factor link to which the map is related.
LABEL_IND	VARCHAR(1)	NULL	Indicates whether the child factor being linked to the parent factor contributes to the label of the parent factor in an Oracle Label Security integration. Possible values are: • Y (for Yes) • N (for No)

Related Topics

- DBA_DV_FACTOR View
  The DBA_DV_FACTOR data dictionary view lists the existing factors in the current PDB.
- DBA_DV_IDENTITY View
  The DBA_DV_IDENTITY data dictionary view lists the identities for each factor.

24.18 DBA_DV_JOB_AUTH View

The DBA_DV_JOB_AUTH data dictionary view lists the authorizations for using Oracle Scheduler in an Oracle Database Vault environment.

For example:

```
SELECT * FROM DBA_DV_JOB_AUTH WHERE GRANTEE = 'PRESTON';
```

Output similar to the following appears:

```
GRANTEE SCHEMA
------- ------
PRESTON OE
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Oracle Scheduler authorization
SCHEMA	VARCHAR2(128)	NOT NULL	Name of the schema on which the user GRANTEE is authorized to perform Oracle Scheduler operations

24.19 DBA_DV_MAC_POLICY View

The DBA_DV_MAC_POLICY data dictionary view lists the Oracle Label Security policies defined for use with Oracle Database Vault.

For example:

```
SELECT POLICY_NAME, ALGORITHM_CODE, ALGORITHM_MEANING
 FROM DBA_DV_MAC_POLICY;
```

Output similar to the following appears:

```
POLICY_NAME     ALGORITHM_CODE    ALGORITHM_MEANING
--------------- ----------------- --------------------------------
ACCESS_DATA     LUI               Minimum Level/Union/Intersection
```

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(128)	NOT NULL	Name of the policy.
ALGORITHM_CODE	VARCHAR(128)	NOT NULL	Merge algorithm code used for the policy. See Related Topics.
ALGORITHM_MEANING	VARCHAR(4000)	NULL	Provides a text description for the corresponding value in the ALGORITHM_CODE column. See Related Topics.
ERROR_LABEL	VARCHAR(4000)	NULL	Label specified for initialization errors, to be set when a configuration error or run-time error occurs during session initialization.

Related Topics

- Table 19-2

- DBA_DV_MAC_POLICY_FACTOR View
  The DBA_DV_MAC_POLICY data dictionary view lists the factors that are associated with Oracle Label Security policies.
- DBA_DV_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.

24.20 DBA_DV_MAC_POLICY_FACTOR View

The DBA_DV_MAC_POLICY data dictionary view lists the factors that are associated with Oracle Label Security policies.

You can use this view to determine what factors contribute to the maximum session label for each policy using the DBA_DV_MAC_POLICY view.

For example:

```
SELECT * FROM DBA_DV_MAC_POLICY_FACTOR;
```

Output similar to the following appears:

```
FACTOR_NAME    MAC_POLICY_NAME
-------------- ------------------
App_Host_Name  Access Locations
```

Column	Datatype	Null	Description
FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the factor
MAC_POLICY_NAME	VARCHAR(128)	NOT NULL	Name of the Oracle Label Security policy associated with this factor

Related Topics

- DBA_DV_MAC_POLICY View
  The DBA_DV_MAC_POLICY data dictionary view lists the Oracle Label Security policies defined for use with Oracle Database Vault.
- DBA_DV_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.

24.21 DBA_DV_MAINTENANCE_AUTH View

The DBA_DV_MAINTENANCE_AUTH data dictionary view provides information about the configuration of Oracle Database Vault authorizations to use Information Life Management (ILM) features.

For example:

```
SELECT GRANTEE, ACTION STATE FROM DBA_DV_MAINTENANCE_AUTH;
```

Output similar to the following appears:

```
GRANTEE                  ACTION
------------------------ --------
PSMITH                   ILM
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR(128)	NOT NULL	Name of the grantee
SCHEMA	VARCHAR(128)	NOT NULL	Schema name or % (for all schemas)
OBJECT	VARCHAR(128)	NOT NULL	Object name or % (for all objects in a schema)
OBJECT_TYPE	VARCHAR(30)	NOT NULL	Object type
ACTION	VARCHAR(30)	NOT NULL	Maintenance action ILM for ILM operations

24.22 DBA_DV_ORADEBUG View

The DBA_DV_ORADEBUG data dictionary view indicates whether users can use the ORADEBUG utility in an Oracle Database Vault environment.

For example:

```
SELECT * FROM DBA_DV_ORADEBUG;
```

Output similar to the following appears:

```
STATE
--------
DISABLED
```

Column	Datatype	Null	Description
STATE	VARCHAR2(8)	NOT NULL	Describes whether the ORADEBUG utility can be used in a Database Vault-enabled environment. Possible values are: • ENABLED means that users can run the ORADEBUG utility • DISABLED means that users cannot run the ORADEBUG utility

24.23 DBA_DV_PATCH_ADMIN_AUDIT View

The DBA_DV_PATCH_ADMIN_AUDIT data dictionary view indicates if auditing has been enabled or disabled for the user who has been granted the DV_ADMIN_PATCH role.

The DBMS_MACADM.ENABLE_DV_PATCH_ADMIN_AUDIT procedure enables this type of auditing.

For example:

```
SELECT * FROM DBA_DV_PATCH_ADMIN_AUDIT;
```

Output similar to the following appears:

```
STATE
--------
DISABLED
```

Column	Datatype	Null	Description
STATE	VARCHAR2(8)	NOT NULL	Describes whether auditing has been enabled or disabled for the DV_ADMIN_PATCH role user. Possible values are: • ENABLED means that the auditing has been enabled • DISABLED means that the auditing has been disabled

Related Topics

- ENABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The ENABLE_DV_PATCH_ADMIN_AUDIT procedure enables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.
- DISABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The DISABLE_DV_PATCH_ADMIN_AUDIT procedure disables realm, command rule, and rule set auditing of the actions by users who have the DV_PATCH_ADMIN role.

24.24 DBA_DV_POLICY View

The DBA_DV_POLICY data dictionary view lists the Oracle Database Vault policies that were created in the current database instance.

For example:

```
SELECT POLICY_NAME, STATE FROM DBA_DV_POLICY
  WHERE STATE = 'ENABLED';
```

Output similar to the following appears:

```
POLICY_NAME                      STATE
-------------------------------- -------
Oracle Account Management Controls ENABLED
Oracle System Protection Controls  ENABLED
```

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(128)	NOT NULL	Names of the Oracle Database Vault policies that have been created.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the policy that was created
STATE	VARCHAR(8)	NULL	Specifies whether the policy is enabled. Possible values are: • ENABLED • DISABLED • SIMULATION
ID#	VARCHAR(1)	NOT NULL	Is a system-generated ID that was assigned to the policy when the policy was created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the policy is a default Oracle Database Vault policy
PL_SQL_STACK	VARCHAR(3)	NULL	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

Related Topics

- DBA_DV_POLICY_OBJECT View
  The DBA_DV_POLICY_OBJECT data dictionary view lists information about the objects that are protected by Oracle Database Vault policies in the current database instance.
- DBA_DV_SIMULATION_LOG View
  The DBA_DV_SIMULATION_LOG data dictionary view captures simulation log information for realms and command rules that have had simulation mode enabled.
- DVSYS.POLICY_OWNER_POLICY View
  The DVSYS.POLICY_OWNER_POLICY data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners.

24.25 DBA_DV_POLICY_LABEL View

The DBA_DV_POLICY_LABEL data dictionary view lists the Oracle Label Security label for each factor identifier in the DBA_DV_IDENTITY view for each policy.

For example:

```
SELECT * FROM DBA_DV_POLICY_LABEL;
```

Output similar to the following appears:

```
IDENTITY_VALUE    FACTOR_NAME     POLICY_NAME       LABEL
----------------- --------------  ----------------  ---------
App_Host_Name     Sect2_Fin_Apps  Access Locations  Sensitive
```

Column	Datatype	Null	Description
IDENTITY_VALUE	VARCHAR(1024)	NOT NULL	Name of the factor identifier.
FACTOR_NAME	VARCHAR(128)	NOT NULL	Name of the factor associated with the factor identifier.
POLICY_NAME	VARCHAR(128)	NOT NULL	Name of the Oracle Label Security policy associated with this factor.
LABEL	VARCHAR(4000)	NOT NULL	Name of the Oracle Label Security label associated with the policy.

Related Topics

- DBA_DV_MAC_POLICY View
  The DBA_DV_MAC_POLICY data dictionary view lists the Oracle Label Security policies defined for use with Oracle Database Vault.
- DBA_DV_MAC_POLICY_FACTOR View
  The DBA_DV_MAC_POLICY data dictionary view lists the factors that are associated with Oracle Label Security policies.

24.26 DBA_DV_POLICY_OBJECT View

The DBA_DV_POLICY_OBJECT data dictionary view lists information about the objects that are protected by Oracle Database Vault policies in the current database instance.

For example:

```
SELECT POLICY_NAME, OBJECT_TYPE FROM DBA_DV_POLICY_OBJECT WHERE POLICY_NAME LIKE
'%Protection Controls';
```

Output similar to the following appears:

```
POLICY_NAME                       OBJECT_TYPE
--------------------------------- ------------
Oracle System Protection Controls  REALM
```

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(128)	NOT NULL	Names of the Oracle Database Vault policies that have been created.
OBJECT_TYPE	VARCHAR(12)	NULL	Type of object that is being protected, such as REALM
COMMAND	VARCHAR(128)	NULL	Name of the command rules that are protected by Database Vault policies
COMMAND_OBJ_OWNER	VARCHAR(128)	NULL	Names of object owners that are associated with Database Vault policies
COMMAND_OBJ_NAME	VARCHAR(128)	NULL	Names of objects that are associated with Database Vault policies
COMMAND_CLAUSE	VARCHAR(100)	NULL	A clause from either the ALTER SYSTEM or ALTER SESSION SQL statement, which was used to create the command rule. For example, you it could list the SET clause for the ALTER SESSION statement. The command rule settings for these two statements are described in the DBMS_MACADM.CREATE_COMMAND_RULE procedure. See Related Topics.
COMMAND_PARAMETER	VARCHAR(128)	NULL	A parameter from the ALTER SYSTEM or ALTER SESSIONcommand rule CLAUSE_NAME setting. See Related Topics.
COMMAND_EVENT	VARCHAR(128)	NULL	An event that the ALTER SYSTEM or ALTER SESSION command rule defines. See Related Topics.
COMMAND_COMPONENT	VARCHAR(128)	NULL	A component of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule. See Related Topics.
COMMAND_ACTION	VARCHAR(128)	NULL	An action of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule. See Related Topics.
COMMON	VARCHAR(3)	NULL	Indicates if the policy objects are local or common. Possible values are: • YES if the policy objects are common • NO if the policy objects are local

Column	Datatype	Null	Description
INHERITED	VARCHAR(3)	NULL	Shows the inheritance status of the policy object, when the COMMON column output is YES. Values are as follows: • YES means that the policy object was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the policy object is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.

Related Topics

- CREATE_COMMAND_RULE Procedure
  The CREATE_COMMAND_RULE procedure creates both command and local command rules, which can be added to a rule set.
- DBA_DV_POLICY View
  The DBA_DV_POLICY data dictionary view lists the Oracle Database Vault policies that were created in the current database instance.
- DBA_DV_POLICY_OWNER View
  The DBA_DV_POLICY_OWNER data dictionary view lists the owners of Oracle Database Vault policies that were created in the current database instance.

24.27 DBA_DV_POLICY_OWNER View

The DBA_DV_POLICY_OWNER data dictionary view lists the owners of Oracle Database Vault policies that were created in the current database instance.

For example:

```
SELECT * FROM DBA_DV_POLICY_OWNER;
```

Output similar to the following appears:

```
POLICY_OWNER                     POLICY_OWNER
-------------------------------- ------------
Oracle System Protection Controls  PSMITH
```

Column	Datatype	Null	Description
POLICY_NAME	VARCHAR(128)	NOT NULL	Names of the Oracle Database Vault policies that have been created.
POLICY_OWNER	VARCHAR(128)	NOT NULL	Names of users who have own Database Vault policies

Related Topics

- DBA_DV_POLICY View
  The DBA_DV_POLICY data dictionary view lists the Oracle Database Vault policies that were created in the current database instance.

- DBA_DV_POLICY_OBJECT View
  The DBA_DV_POLICY_OBJECT data dictionary view lists information about the objects that are protected by Oracle Database Vault policies in the current database instance.

24.28 DBA_DV_PREPROCESSOR_AUTH View

The DBA_DV_PREPROCESSOR_AUTH data dictionary view shows users who have been granted authorization to run preprocessor programs through external tables.

For example:

```
SELECT * FROM DBA_DV_PREPROCESSOR_AUTH WHERE GRANTEE = 'PFITCH';
```

Output similar to the following appears:

```
GRANTEE
-------
PFITCH
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted authorization to run preprocessor programs

Related Topics

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

24.29 DBA_DV_PROXY_AUTH View

The DBA_DV_PROXY_AUTH data dictionary view lists the proxy users and schemas that were specified by the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure.

This procedure grants a proxy user authorization to proxy other user accounts.

For example:

```
SELECT * FROM DBA_DV_DDL_AUTH WHERE GRANTEE = 'PRESTON';
```

Output similar to the following appears:

```
GRANTEE SCHEMA
------- ------
PRESTON DKENT
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the proxy user
SCHEMA	VARCHAR2(128)	NOT NULL	Name of the schema that is proxied by the GRANTEE user.

Related Topics

- AUTHORIZE_PROXY_USER Procedure
  The AUTHORIZE_PROXY_USER procedure grants a proxy user authorization to proxy other user accounts, as long as the proxy user has database authorization.
- UNAUTHORIZE_PROXY_USER Procedure
  The UNAUTHORIZE_PROXY_USER procedure revokes authorization from a user who was granted proxy authorization from the DBMS_MACADM.AUTHORIZE_PROXY_USER procedure.

24.30 DBA_DV_PUB_PRIVS View

The DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in Oracle Database Vault Administrator.

See also Privilege Management - Summary Reports.

For example:

```
SELECT USERNAME, ACCESS_TYPE FROM DBA_DV_PUB_PRIVS WHERE USERNAME = 'OE';
```

Output similar to the following appears:

```
USERNAME    ACCESS_TYPE
----------- -----------------
OE          PUBLIC
```

Column	Datatype	Null	Description
USERNAME	VARCHAR(128)	NOT NULL	Database schema in the current database instance.
ACCESS_TYPE	VARCHAR(128)	NULL	Access type granted to the user listed in the USERNAME column (for example, PUBLIC).
PRIVILEGE	VARCHAR(40)	NOT NULL	Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(128)	NOT NULL	Owner of the database schema to which the USERNAME user has been granted privileges.
OBJECT_NAME	VARCHAR(128)	NOT NULL	Name of the object within the schema listed in the OWNER column.

Related Topics

- Privilege Management - Summary Reports
  The privilege management summary reports track privilege distribution by grantees, owners, and privileges.
- DBA_DV_USER_PRIVS View
  The DBA_DV_USER_PRIVS data dictionary view lists the privileges for a database user account excluding privileges granted through the PUBLIC role.
- DBA_DV_USER_PRIVS_ALL View
  The DBA_DV_USER_PRIVS_ALL data dictionary view lists the privileges for a database account including privileges granted through PUBLIC.
- DBA_DV_ROLE View
  The DBA_DV_ROLE data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.

24.31 DBA_DV_REALM View

The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.

For example:

```
SELECT NAME, ENABLED, COMMON FROM DBA_DV_REALM ORDER BY NAME;
```

Output similar to the following appears:

```
NAME                                ENABLED  COMMON
---------------------------------  -------- ------
Database Vault Account Management         Y     NO
...
```

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Names of the realms created.
DESCRIPTION	VARCHAR(1024)	NOT NULL	Description of the realm created.
AUDIT_OPTIONS	NUMBER	NOT NULL	Specifies whether auditing using traditional 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. Starting with Oracle Database 23c, traditional auditing is desupported.
REALM_TYPE	VARCHAR(9)	NULL	Type of realm: whether it is a regular realm or a mandatory realm. See realm_type in the UPDATE_REALM command description for more information about possible values. (See Related Topics.)
COMMON	VARCHAR(3)	NOT NULL	Indicates whether the realm is local or common. Possible values are: • YES if the realm is common • NO if the realm is local

Column	Datatype	Null	Description
INHERITED	VARCHAR(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ENABLED	VARCHAR(1)	NOT NULL	Possible values are as follows: • Y indicates that realm checking is enabled • N indicates it is disabled • S indicates the realm is in simulation mode
ID#	NUMBER	NOT NULL	The ID number of the realm, which is automatically generated when the realm is created
ORACLE_SUPPLIED	VARCHAR(3)	NOT NULL	Indicates whether the realm is a default (that is, Oracle-supplied) realm or a user-created command rule. Possible values are: • YES if the realm is a default realm • NO if the realm is a user-created realm
PL_SQL_STACK	VARCHAR(3)	NULL	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

Related Topics

- DBA_DV_REALM_AUTH View
  The DBA_DV_REALM_AUTH data dictionary view lists database user account or role authorization (GRANTEE) who can access realm objects.
- DBA_DV_REALM_OBJECT View
  The DBA_DV_REALM_OBJECT data dictionary view lists the database schemas, or subsets of schemas, that are secured by the realms.
- UPDATE_REALM Procedure
  The UPDATE_REALM procedure updates a realm.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

24.32 DBA_DV_REALM_AUTH View

The DBA_DV_REALM_AUTH data dictionary view lists database user account or role authorization (GRANTEE) who can access realm objects.

For example:

```
SELECT REALM_NAME, GRANTEE, AUTH_RULE_SET_NAME FROM DBA_DV_REALM_AUTH;
```

Output similar to the following appears:

```
REALM_NAME                    GRANTEE  AUTH_RULE_SET_NAME
----------------------------- -------- --------------------
Performance Statistics Realm  SYSADM   Check Conf Access
```

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(128)	NULL	Name of the realm.
COMMON_REALM	VARCHAR(3)	NULL	For a multitenant environment, indicates whether the realm is local or common. Possible values are: • YES if the realm is common • NO if the realm is local
INHERITED_REALM	VARCHAR(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
GRANTEE	VARCHAR(128)	NOT NULL	User or role name to authorize as owner or participant.
AUTH_RULE_SET_NAME	VARCHAR(128)	NULL	Rule set to check before authorizing. If the rule set evaluates to TRUE, then the authorization is allowed.
AUTH_OPTIONS	VARCHAR(4000)	NULL	Type of realm authorization: either Participant or Owner.
COMMON_AUTH	VARCHAR(3)	NULL	Indicates whether the authorization to the common realm is local or common. Possible values are: • YES if the authorization is common • NO if the authorization is local to this PDB

Column	Datatype	Null	Description
INHERITED_AUTH	VARCHAR(3)	NULL	Shows the inheritance status of the realm authorization, when the COMMON_AUTH column output is YES. Values are as follows: • YES means that the realm authorization was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was applied. • NO means that the realm authorization is local, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED_AUTH value NO but a CDB root common command rule will have an INHERITED_AUTH value of YES.

Related Topics

- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.
- DBA_DV_REALM View
  The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.
- DBA_DV_REALM_OBJECT View
  The DBA_DV_REALM_OBJECT data dictionary view lists the database schemas, or subsets of schemas, that are secured by the realms.

24.33 DBA_DV_REALM_OBJECT View

The DBA_DV_REALM_OBJECT data dictionary view lists the database schemas, or subsets of schemas, that are secured by the realms.

See About Realm-Secured Objects for more information.

For example:

```
SELECT REALM_NAME, OWNER, OBJECT_NAME, COMMON_REALM FROM DBA_DV_REALM_OBJECT;
```

Output similar to the following appears:

```
REALM_NAME                    OWNER    OBJECT_NAME COMMON_REALM
----------------------------- -------- ----------- ------------
Performance Statistics Realm OE       ORDERS      NO
```

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(128)	NOT NULL	Name of the realm.
COMMON_REALM	VARCHAR(3)	NOT NULL	Indicates whether this realm is a common realm or a local realm. Possible values are: • YES if the realm is common • NO if the realm is local

Column	Datatype	Null	Description
INHERITED_REALM	VARCHAR(3)	NOT NULL	Shows the inheritance status of the realm when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
OWNER	VARCHAR(128)	NOT NULL	Database schema owner who owns the object.
OBJECT_NAME	VARCHAR(128)	NOT NULL	Name of the object the realm protects.
OBJECT_TYPE	VARCHAR(32)	NOT NULL	Type of object the realm protects, such as a database table, view, index, or role.

Related Topics

- About Realm-Secured Objects
  Realm-secured objects define the territory—a set of schema and database objects and roles—that a realm protects.
- DBA_DV_REALM View
  The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.
- DBA_DV_REALM_AUTH View
  The DBA_DV_REALM_AUTH data dictionary view lists database user account or role authorization (GRANTEE) who can access realm objects.

24.34 DBA_DV_ROLE View

The DBA_DV_ROLE data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.

For example:

```
SELECT ROLE, RULE_NAME FROM DBA_DV_ROLE;
```

Output similar to the following appears:

```
ROLE                 RULE_NAME
-------------------- --------------------
Sector2_APP_MGR      Check App2 Access
Sector2_APP_DBA      Check App2 Access
```

Column	Datatype	Null	Description
ROLE	VARCHAR(128)	NOT NULL	Name of the secure application role.

Column	Datatype	Null	Description
RULE_NAME	VARCHAR(128)	NOT NULL	Name of the rule set associated with the secure application role.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the secure application role is enabled. Possible values are: • Y (Yes) if the role is enabled • N (No) if the role is disabled
ID#	NUMBER	NOT NULL	The ID number of the command rule, which is automatically generated when the command rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NOT NULL	Indicates whether the command rule is a default (that is, Oracle-supplied) command rule or a user-created command rule. Possible values are: • YES if the command rule is a default command rule • NO if the command rule is a user-created command rule

Related Topics

- DBA_DV_PUB_PRIVS View
  The DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in Oracle Database Vault Administrator.
- DBA_DV_USER_PRIVS View
  The DBA_DV_USER_PRIVS data dictionary view lists the privileges for a database user account excluding privileges granted through the PUBLIC role.
- DBA_DV_USER_PRIVS_ALL View
  The DBA_DV_USER_PRIVS_ALL data dictionary view lists the privileges for a database account including privileges granted through PUBLIC.

24.35 DBA_DV_RULE View

The DBA_DV_RULE data dictionary view lists the rules that have been defined.

For example:

```
SELECT NAME, RULE_EXPR FROM DBA_DV_RULE WHERE NAME = 'Maintenance Window';
```

Output similar to the following appears:

```
NAME               RULE_EXP
------------------ ----------------------------------------------
Maintenance Window  TO_CHAR(SYSDATE,'HH24') BETWEEN '10' AND '12'
```

To find the rule sets that use specific rules, query the DBA_DV_RULE_SET_RULE view.

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression for the rule.

Column	Datatype	Null	Description
COMMON	VARCHAR(3)	NOT NULL	Indicates whether the rule is local or common. Possible values are: • YES if the rule is common • NO if the rule is local
INHERITED	VARCHAR(3)	NULL	Shows the inheritance status of the rule, when the COMMON column output is YES. Values are as follows: • YES means that the rule was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the rule is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ID#	NUMBER	NOT NULL	The ID number of the rule, which is automatically generated when the rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the rule is a default (that is, Oracle-supplied) rule or a user-created rule. Possible values are: • YES if the rule is a default rule • NO if the rule is a user-created rule
IS_STATIC	VARCHAR(5)	NULL	Indicates whether the evaluation of a rule is static or dynamic. Possible values are: • TRUE means the rule is static. The result of the first rule evaluation in a session is cached and used whenever the rule needs to be evaluated again in the session. • FALSE (default) means the rule is not static, and hence not cached for use later in the session.

Related Topics

- DBA_DV_RULE_SET View
  The DBA_DV_RULE_SET data dictionary view lists the rules sets that have been created.
- DBA_DV_RULE_SET_RULE View
  The DBA_DV_RULE_SET_RULE data dictionary view lists rules that are associated with existing rule sets.

24.36 DBA_DV_RULE_SET View

The DBA_DV_RULE_SET data dictionary view lists the rules sets that have been created.

For example:

```
SELECT RULE_SET_NAME, HANDLER_OPTIONS, HANDLER FROM DBA_DV_RULE_SET
 WHERE RULE_SET_NAME = 'Maintenance Period';
```

Output similar to the following appears:

```
RULE_SET_NAME       HANDLER_OPTIONS  HANDLER
------------------- ---------------- ----------------------
Maintenance Period                   1 dbavowner.email_alert
```

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the rule set.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the rule set has been enabled. Y (Yes) enables the rule set; N (No) disables it.
EVAL_OPTIONS_MEANING	VARCHAR(4000)	NULL	For rules sets that contain multiple rules, determines how many rules are evaluated. Possible values are: • All True: All rules in the rule set must evaluate to true for the rule set itself to evaluate to TRUE. • Any True: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to TRUE.
AUDIT_OPTIONS	NUMBER	NOT NULL	Indicates when auditing using traditional auditing is used. Possible values are: • 0: No auditing • 1: Audit on failure • 2: Audit on success • 3: Audit on both failure and success Starting with Oracle Database 23c, traditional auditing is desupported.
FAIL_OPTIONS_MEANING	VARCHAR(4000)	NULL	Determines when an audit record is created for the rule set. Possible values are: • Do Not Show Error Message. • Show Error Message
FAIL_MESSAGE	VARCHAR(80)	NULL	Error message for failure that is associated with the fail code listed in the FAIL_CODE column.
FAIL_CODE	VARCHAR(10)	NULL	The error message number associated with the message listed in the FAIL_MESSAGE column. Possible values are in the ranges of -20000 to -20999 or 20000 to 20999.
HANDLER_OPTIONS	NUMBER	NOT NULL	Determines how error handling is used. Possible values are: • 0: Disables error handling. • 1: Call handler on rule set failure. • 2: Call handler on rule set success.
HANDLER	VARCHAR(1024)	NULL	Name of the PL/SQL function or procedure that defines the custom event handler logic.

Column	Datatype	Null	Description
IS_STATIC	VARCHAR2(5)	NULL	Indicates how often the rule set is evaluated during a user session. Possible values are: • TRUE: The rule set is evaluated once, and result of the rule set is reused throughout the user session. • FALSE (default): The rule set is evaluated each time it is accessed during the user session.
COMMON	VARCHAR2(3)	NULL	Indicates whether the rule set is local or common. Possible values are: • YES if the rule set is common • NO if the rule set is local
INHERITED	VARCHAR2(3)	NULL	Shows the inheritance status of the rule set, when the COMMON column output is YES. Values are as follows: • YES means that the rule set was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the rule set is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ID#	NUMBER)	NOT NULL	The ID number of the rule set, which is automatically generated when the rule set is created
ORACLE_SUPPLIED	VARCHAR2(3)	NULL	Indicates whether the rule set is a default (that is, Oracle-supplied) rule set or a user-created rule set. Possible values are: • YES if the rule set is a default rule set • NO if the rule set is a user-created rule set

Related Topics

- DBA_DV_RULE View
  The DBA_DV_RULE data dictionary view lists the rules that have been defined.
- DBA_DV_RULE_SET_RULE View
  The DBA_DV_RULE_SET_RULE data dictionary view lists rules that are associated with existing rule sets.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

24.37 DBA_DV_RULE_SET_RULE View

The DBA_DV_RULE_SET_RULE data dictionary view lists rules that are associated with existing rule sets.

For example:

```
SELECT RULE_SET_NAME, RULE_NAME, RULE_EXPR FROM DBA_DV_RULE_SET_RULE
 WHERE RULE_NAME = 'Is Security Officer';
```

Output similar to the following appears:

```
RULE_SET_NAME                RULE_NAME          RULE_EXP
---------------------------- ------------------ --------------------------------
Can Grant VPD Administration Is Security Owner  DBMS_MACUTL.USER_HAS_ROLE_VARCHAR
                                                ('DV_OWNER',
                                                  dvsys.dv_login_user) = 'Y'
```

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set that contains the rule.
RULE_NAME	VARCHAR(128)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression that defines the rule listed in the RULE_NAME column.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the rule is enabled or disabled. Y (Yes) enables the rule set; N (No) disables it.
RULE_ORDER	NUMBER	NOT NULL	The order in which rules are used within the rule set. Does not apply to this release.
COMMON	VARCHAR(3)	NOT NULL	Indicates whether the rule is local or common. Possible values are: • YES if the rule is common • NO if the rule is local
INHERITED	VARCHAR(3)	NOT NULL	Shows the inheritance status of the rule, when the COMMON column output is YES. Values are as follows: • YES means that the rule was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the rule is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.

Related Topics

- DBA_DV_RULE View
  The DBA_DV_RULE data dictionary view lists the rules that have been defined.

- DBA_DV_RULE_SET View
  The DBA_DV_RULE_SET data dictionary view lists the rules sets that have been created.

24.38 DBA_DV_SIMULATION_LOG View

The DBA_DV_SIMULATION_LOG data dictionary view captures simulation log information for realms and command rules that have had simulation mode enabled.

For example:

```
SELECT USERNAME, COMMAND
FROM DBA_DV_SIMULATION_LOG, REALM_NAME
WHERE REALM_NAME = 'HR Realm';
```

Output similar to the following appears:

```
USERNAME      COMMAND
------------- --------------------------------------------
PSMITH        SELECT
```

Column	Datatype	Null	Description
ID	NUMBER	NOT NULL	Simulation log ID
USERNAME	VARCHAR2(128)	NOT NULL	Name of the user whose information is being tracked
COMMAND	VARCHAR2(128)	NOT NULL	Command rule being tracked For a listing of existing command rules, query the DBA_DV_COMMAND_RULE view.
VIOLATION_TYPE	VARCHAR2(4000)	NULL	Type of violation. See Table 24-2 for more information.
REALM_NAME	VARCHAR2(4000)	NULL	Realm being tracked. Multiple realms are represented as comma separated names in the VARCHAR2 field. For a listing of existing realms, query the DBA_DV_REALM view.
REALM_TYPE	VARCHAR2(9)	NULL	Type of realm being tracked (for example, mandatory realms).
OBJECT_OWNER	VARCHAR2(128)	NULL	For command rules, the database schema to which the command rule applied
OBJECT_NAME	VARCHAR2(128)	NULL	For command rules, the database object that the command rule protects
OBJECT_TYPE	VARCHAR2(129)	NULL	For command rules, the type of object that is being protected
RULE_SET_NAME	VARCHAR2(4000)	NULL	Rule set being tracked; it is associated with a command rule. Multiple rule sets are represented as comma separated names in the VARCHAR2 field. For a listing of existing rule sets, query the DBA_DV_RULE_SET view, described in DBA_DV_RULE_SET View

Column	Datatype	Null	Description
RETURNCODE	NUMBER	NOT NULL	The Oracle Database ORA error that results if the Database Vault entity was in the enabled state rather than in simulation state
SQLTEXT	VARCHAR2(4000)	NULL	SQL text that the simulation mode captures
AUTHENTICATION_METHOD	VARCHAR2(10)	NULL	Authentication method used.
CLIENT_IP	VARCHAR2(45)	NULL	The IP address of the machine from which the client is connected
DB_DOMAIN	VARCHAR2(128)	NULL	The domain of the database as specified in the DB_DOMAIN initialization parameter
DATABASE_HOSTNAME	VARCHAR2(128)	NULL	The host name of the computer on which the instance is running
DATABASE_INSTANCE	VARCHAR2(5)	NULL	The instance identification number of the current instance
DATABASE_IP	VARCHAR2(45)	NULL	The IP address of the computer on which the instance is running
DATABASE_NAME	VARCHAR2(128)	NULL	The name of the database as specified in the DB_NAME initialization parameter
DOMAIN	VARCHAR2(4000)	NULL	A named collection of physical, configuration, or implementation-specific factors in the run-time environment.
ENTERPRISE_IDENTITY	VARCHAR2(1024)	NULL	The enterprise-wide identity for the user.
IDENTIFICATION_TYPE	VARCHAR2(14)	NULL	The way the user schema was created in the database.
LANG	VARCHAR2(10)	NULL	The ISO abbreviation for the language name, a shorter form than the existing LANGUAGE parameter
LANGUAGE	VARCHAR2(100)	NULL	The language and territory your session currently uses, along with the database character set.
MACHINE	VARCHAR2(64)	NULL	The host name for the database client that established the current session. If you must find out whether the computer was used for a client or server session, then you can compare this setting with the Database_Hostname factor to make the determination
NETWORK_PROTOCOL	VARCHAR2(4)	NULL	The network protocol being used for communication, as specified in the PROTOCOL=protocol portion of the connect string
PROXY_ENTERPRISE_IDENTITY	VARCHAR2(1024)	NULL	The Oracle Internet Directory DN when the proxy user is an enterprise user
PROXY_USER	VARCHAR2(128)	NULL	The name of the database user who opened the current session on behalf of SESSION_USER

Column	Datatype	Null	Description
SESSION_USER	VARCHAR2(128)	NULL	The database user name by which the current user is authenticated. This value remains the same throughout the session.
DV$_DBLINK_INFO	VARCHAR2(128)	NULL	Returns the source of a database link session. The string that it returns has this form: SOURCE_GLOBAL_NAME=*dblink_src_global_name*, DBLINK_NAME=*dblink_name*, SOURCE_AUDIT_SESSIONID=*dblink_src_audit_sessionid* In this specification: • *dblink_src_global_name* is the unique global name of the source database • *dblink_name* is the name of the database link on the source database • *dblink_src_audit_sessionid* source database that initiated source database that initiated the connection to the remote database using *dblink_name*
DV$_MODULE	VARCHAR2(64)	NULL	The application name (module) that was set through the DBMS_APPLICATION_INFO PL/SQL package or Oracle Call Interface (OCI).
DV$_CLIENT_IDENTIFIER	VARCHAR2(64)	NULL	Returns an identifier that is set by the application through the DBMS_SESSION.SET_IDENTIFIER procedure, the OCI attribute OCI_ATTR_CLIENT_IDENTIFIER, or Oracle Dynamic Monitoring Service (DMS). Various Oracle Database components use this attribute to identify lightweight application users who authenticate as the same database user.
FACTOR_CONTEXT	VARCHAR2(4000)	NULL	An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered
TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	NULL	Time stamp of user action, in UTC (Coordinated Universal Time) time zone
PL_SQL_STACK	CLOB	NULL	When simulation mode is enabled, indicates whether the PL/SQL stack has been recorded for failed operations. TRUE indicates that the PL/SQL stack has been recorded; FALSE indicates that the PL/SQL stack has not been recorded.

VIOLATION_TYPE Code Values

Table 24-2 lists the VIOLATION_TYPE code values for the DBA_DV_SIMULATION_LOG view.

Table 24-2 DBA_DV_SIMULATION_LOG VIOLATION_TYPE Code Values

Code	Meaning
1000	Realm violation
1001	Command rule violation
1002	Oracle Data Pump authorization violation
1003	Simulation violation
1004	Oracle Scheduler authorization violation
1005	DDL authorization violation
1006	PARSE_AS_USER violation

Related Topics

- DBA_DV_REALM View
  The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.
- DBA_DV_COMMAND_RULE View
  The DBA_DV_COMMAND_RULE data dictionary view lists the SQL statements that are protected by command rules.
- DBA_DV_POLICY View
  The DBA_DV_POLICY data dictionary view lists the Oracle Database Vault policies that were created in the current database instance.

24.39 DBA_DV_STATUS or SYS.DBA_DV_STATUS View

The DBA_DV_STATUS (or SYS.DBA_DV_STATUS) data dictionary view shows the status of Oracle Database Vault being enabled and configured.

For example:

```
SELECT * FROM DBA_DV_STATUS;
```

Output similar to the following appears:

```
NAME                 STATUS
-------------------- --------------
DV_APP_PROTECTION    NOT CONFIGURED
DV_CONFIGURE_STATUS  TRUE
DV_ENABLE_STATUS     TRUE
```

Related Views

- CDB_DV_STATUS View

Column	Datatype	Null	Description
NAME	VARCHAR2(19)	NOT NULL	Shows one of the following settings: • DV_APP_PROTECTION shows whether Database Vault operations control has been configured or not configured • DV_CONFIGURE_STATUS shows whether Oracle Database Vault has been configured, that is, with the CONFIGURE_DV procedure. • DV_ENABLE_STATUS shows whether Oracle Database Vault has been enabled, that is, with the DBMS_MACADM.ENABLE_DV procedure.
STATUS	VARCHAR2(64)	NOT NULL	TRUE means that Oracle Database Vault is configured or enabled; FALSE means that it is not. For DV_APP_PROTECTION, it shows either CONFIGURED or NOT CONFIGURED.

24.40 DBA_DV_SQL_FIREWALL_AUTH View

The DBA_DV_SQL_FIREWALL_AUTH data dictionary view shows users who have been granted authorization to perform SQL Firewall operations.

For example:

```
SELECT * FROM DBA_DV_SQL_FIREWALL_AUTH WHERE GRANTEE = 'PFITCH';
```

Output similar to the following appears:

```
GRANTEE  CAN_MANAGE_DV_ADMINS
-------  --------------------
PFITCH   Y
```

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NOT NULL	Name of the user who has been granted Database Replay workload capture authorization
CAN_MANAGE_DV_ADMINS	VARCHAR2(32)	NOT NULL	Setting for the AUTHORIZE_SQL_FIREWALL manage_dv_admins parameter for this user

Related Topics

- Using SQL Firewall with Oracle Database Vault
  You can authorize SQL Firewall users to work in a Database Vault environment.

24.41 DBA_DV_TTS_AUTH View

The DBA_DV_TTS_AUTH data dictionary view lists users who have been granted authorization through the DBMS_MACADM.AUTHORIZE_TTS_USER procedure to perform Oracle Data Pump transportable operations.

For example:

```
SELECT * FROM DBA_DV_TTS_AUTH;
```

Output similar to the following appears:

```
GRANTEE  TSNAME
-------- --------
DB_MGR   HR_TS
```

Related Views

- DBA_DV_DATAPUMP_AUTH View

Column	Datatype	Null	Description
GRANTEE	VARCHAR(128)	NOT NULL	Name of the user who has been granted transportable tablespace authorization
TSNAME	VARCHAR(128)	NOT NULL	Name of the transportable tablespace to which the GRANTEE user has been granted authorization

Related Topics

- Using Oracle Data Pump with Oracle Database Vault
  Database administrators can authorize Oracle Data Pump users to work in a Database Vault environment.
- DBA_DV_DATAPUMP_AUTH View
  The DBA_DV_DATAPUMP_AUTH data dictionary view lists the authorizations for using Oracle Data Pump in an Oracle Database Vault environment.

24.42 DBA_DV_USER_PRIVS View

The DBA_DV_USER_PRIVS data dictionary view lists the privileges for a database user account excluding privileges granted through the PUBLIC role.

For example:

```
SELECT USERNAME, ACCESS_TYPE, PRIVILEGE FROM DBA_DV_USER_PRIVS;
```

Output similar to the following appears:

```
USERNAME  ACCESS_TYPE          PRIVILEGE
--------- -------------------- ------------
DVOWNER   DV_ADMIN             SELECT
SYS       SELECT_CATALOG_ROLE  SELECT
...
```

Column	Datatype	Null	Description
USERNAME	VARCHAR(128)	NOT NULL	Name of the database schema account in which privileges have been defined.
ACCESS_TYPE	VARCHAR(128)	NULL	Role the database user account listed in the USERNAME column uses to access the database. Oracle Database Vault accounts have direct access.
PRIVILEGE	VARCHAR(40)	NOT NULL	Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(128)	NOT NULL	Name of the database user account.
OBJECT_NAME	VARCHAR(128)	NOT NULL	Name of the PL/SQL function or procedure used to define privileges.

Related Topics

- DBA_DV_PUB_PRIVS View
  The DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in Oracle Database Vault Administrator.
- DBA_DV_ROLE View
  The DBA_DV_ROLE data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.
- 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.

24.43 DBA_DV_USER_PRIVS_ALL View

The DBA_DV_USER_PRIVS_ALL data dictionary view lists the privileges for a database account including privileges granted through PUBLIC.

For example:

```
SELECT USERNAME, ACCESS_TYPE, PRIVILEGE FROM DBA_DV_USER_PRIVS;
```

Output similar to the following appears:

```
USERNAME             ACCESS_TYPE  PRIVILEGE
-------------------- ------------ -----------------
BEA_DVACCTMGR        CONNECT      CREATE_SESSION
LEO_DVOWNER          DIRECT       CREATE PROCEDURE
...
```

Column	Datatype	Null	Description
USERNAME	VARCHAR(128)	NULL	Name of the database schema account in which privileges have been defined.
ACCESS_TYPE	VARCHAR(128)	NULL	Role the database user account listed in the USERNAME column uses to access the database. Oracle Database Vault accounts have direct access.
PRIVILEGE	VARCHAR(40)	NULL	Privilege granted to the user listed in the USERNAME column.
OWNER	VARCHAR(128)	NULL	Name of the database user account.
OBJECT_NAME	VARCHAR(128)	NULL	Name of the PL/SQL function or procedure used to define privileges.

Related Topics

- DBA_DV_PUB_PRIVS View
  The DBA_DV_PUB_PRIVS data dictionary view lists data reflected in the Oracle Database Vault privilege management reports used in Oracle Database Vault Administrator.
- DBA_DV_ROLE View
  The DBA_DV_ROLE data dictionary view lists the Oracle Database Vault secure application roles used in privilege management.
- 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.

24.44 DVSYS.DV$CONFIGURATION_AUDIT View

The DVSYS.DV$CONFIGURATION_AUDIT data dictionary view captures DVSYS.AUDIT_TRAIL$ table audit trail records.

It includes records that are related to successful and failed configuration changes made to realms, rules, rule sets, factors, and other Oracle Database Vault policy configuration activities.

For example:

```
SELECT USERNAME, ACTION_NAME FROM DVSYS.DV$CONFIGURATION_AUDIT
WHERE USERNAME = 'PSMITH';
```

Output similar to the following appears:

```
USERNAME   ACTION_NAME
---------- --------------------
PSMITH     Realm Creation Audit
PSMITH     Rule Set Update Audit
```

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR(255)	NULL	Operating system login user name of the user whose actions were audited
USERNAME	VARCHAR(128)	NULL	Name of the database user whose actions were audited
USERHOST	VARCHAR2(128)	NULL	Client computer name
TERMINAL	VARCHAR2(30)	NULL	Identifier for the user's terminal
TIMESTAMP	DATA	NULL	Date and time of creation of the audit trail entry (in the local database session time zone)
OWNER	VARCHAR2(128)	NULL	Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR2(128)	NULL	Name of the object affected by the action. Expected values are: • ROLE$ • REALM$ • CODE$ • FACTOR$
ACTION	NUMBER	NOT NULL	Numeric action type code. The corresponding name of the action type is in the ACTION_NAME column. See Table 24-3 for a listing of the possible actions.
ACTION_NAME	VARCHAR2(128)	NULL	Name of the action type corresponding to the numeric code in the ACTION column. See Table 24-3 for a listing of the possible actions.
ACTION_OBJECT_ID	NUMBER	NULL	The unique identifier of the record in the table specified under OBJ_NAME
ACTION_OBJECT_NAME	VARCHAR2(128)	NULL	The unique name or natural key of the record in the table specified under OBJ_NAME

Column	Datatype	Null	Description
ACTION_COMMAND	VARCHAR2(4000)	NULL	The SQL text of the command procedure that was run that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)	NULL	The labels for all (traditional) 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. Starting with Oracle Database 23c, traditional auditing is desupported.
RULE_SET_ID	NUMBER	NULL	The unique identifier of the rule set that was executing and caused the audit event to trigger
RULE_SET_NAME	VARCHAR2(128)	NULL	The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER	NULL	Not used
RULE_NAME	VARCHAR2(128)	NULL	Not used
FACTOR_CONTEXT	VARCHAR2(4000)	NULL	An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered
COMMENT_TEXT	VARCHAR2(4000)	NULL	Text comment on the audit trail entry, providing more information about the statement audited
SESSIONID	NUMBER	NOT NULL	Numeric identifier for each Oracle session
ENTRYID	NUMBER	NOT NULL	Same as the value in the ID# column
STATEMENTID	NUMBER	NOT NULL	Numeric identifier for the statement invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
RETURNCODE	NUMBER	NOT NULL	Oracle error code generated by the action. The error code for a statement or procedure invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
EXTENDED_TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	NULL	Time stamp of creation of the audit trail entry (time stamp of user login for entries) in UTC (Coordinated Universal Time) time zone
PROXY_SESSIONID	NUMBER	NULL	Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)	NULL	Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER	NULL	Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)	NULL	Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(128)	NULL	Database login user name of the user whose actions were audited
CREATE_DATE	DATE	NULL	Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(128)	NULL	Same as CREATED_BY column value
UPDATE_DATE	DATE	NULL	Same as UPDATED_BY column value

Column	Datatype	Null	Description
GRANTEE	VARCHAR2(128)	NULL	User ID of users who have been granted Database Vault-protected roles, realm authorization, command-rule authorization, job scheduler authorization, or Oracle Data Pump authorizations
ENABLED_STATUS	VARCHAR2(1)	NULL	Indicates whether the configuration was enabled

Table 24-3 describes the possible values for the ACTION column of the DVSYS.DV$CONFIGURATION_AUDIT view.

Table 24-3 DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20001	Enable DV enforcement Audit
20002	Disable DV enforcement Audit
20003	Realm Creation Audit
20004	Realm Update Audit
20005	Realm Rename Audit
20006	Realm Deletion Audit
20007	Add Realm Auth Audit
20008	Delete Realm Auth Audit
20009	Update Realm Auth Audit
20010	Add Realm Object Audit
20011	Update Realm Object Audit
20012	Delete Realm Object Audit
20013	Enable Event Audit
20014	Disable Event Audit
20015	Rule Set Creation Audit
20016	Rule Set Update Audit
20017	Rule Set Rename Audit
20018	Rule Set Deletion Audit
20019	Add Rule To Rule Set Audit
20020	Delete Rule From Rule Set Audit
20021	Rule Creation Audit
20022	Rule Update Audit
20023	Rule Rename Audit
20024	Rule Deletion Audit
20025	CommandRule Creation Audit
20026	CommandRule Update Audit
20027	CommandRule Deletion Audit
20028	Authorize Datapump User Audit

Table 24-3 (Cont.) DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20029	Unauthorize Datapump User Audit
20030	Authorize Job User Audit
20031	Unauthorize Job User Audit
20032	Factor_Type Creation Audit
20033	Factor_Type Deletion Audit
20034	Factor_Type Update Audit
20035	Factor_Type Rename Audit
20036	Factor Creation Audit
20037	G_FACTOR_DELETION_AUDIT_CODE
20038	Factor Update Audit
20039	Factor Rename Audit
20040	Add Factor Link Audit
20041	Delete Factor Link Audit
20042	Add Policy Factor Audit
20043	Delete Policy Factor Audit
20044	Create Identity Audit
20045	Delete Identity Audit
20046	Update Identity Audit
20047	Change Identity Factor Audit
20048	Change Identity Value Audit
20049	Create Identity Map Audit
20050	Delete Identity Map Audit
20051	Create Policy Label Audit
20052	Delete Policy Label Audit
20053	Create Mac Policy Audit
20054	Update Mac Policy Audit
20055	Delete Mac Policy Audit
20056	Create Role Audit
20057	Delete Role Audit
20058	Update Role Audit
20059	Rename Role Audit
20060	Create Domain Identity Audit
20061	Drop Domain Identity Audit
20062	Enable Oradebug Audit
20063	Disable Oradebug Audit
20064	Authorize Proxy User Audit

Table 24-3 (Cont.) DVSYS.DV$CONFIGURATION_AUDIT View ACTION Values

Action Type Code	Action Name
20065	Unauthorize Proxy User Audit
20066	Enable DV Dictionary Accounts Audit
20067	Disable DV Dictionary Accounts Audit
20068	Authorize DDL Audit
20069	Unauthorize DDL Audit
20070	Authorize TTS Audit
20071	Unauthorize TTS Audit
20072	Authorize PREPROCESSOR Audit
20073	Unauthorize PREPROCESSOR Audit
20074	Create Policy Audit
20075	Update Policy Description Audit
20076	Update Policy State Audit
20077	Rename Policy Audit
20078	Drop Policy Audit
20079	Add Realm to Policy Audit
20080	Delete Realm From Policy Audit
20081	Add Command Rule to Policy Audit
20082	Delete Command Rule from Policy Audit
20083	Add Policy Owner Audit
20084	Delete Policy Owner Audit
20085	Authorize Maintenance Audit
20086	Unauthorize Maintenance Audit

Related Topics

- AUDSYS.DV$CONFIGURATION_AUDIT View
  The `AUDSYS.DV$CONFIGURATION_AUDIT` view is almost the same as the `DVSYS.DV$CONFIGURATION_AUDIT` view except that it captures unified audit trail Database Vault audit records.

24.45 DVSYS.DV$ENFORCEMENT_AUDIT View

The `DVSYS.DV$ENFORCEMENT_AUDIT` data dictionary view provides information about enforcement-related audits from the `DVSYS.AUDIT_TRAIL$` table.

It captures user violations on command rules, realms, and factors.

For example:

```
SELECT USERNAME, ACTION_COMMMAND FROM DVSYS.DV$ENFORCEMENT_AUDIT
WHERE OWNER = 'HR';
```

Output similar to the following appears:

```
USERNAME    ACTION_COMMMAND
----------- ------------------------------
PSMITH      CREATE_REALM
```

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR(255)	NULL	Operating system login user name of the user whose actions were audited
USERNAME	VARCHAR(128)	NULL	Name of the database user whose actions were audited
USERHOST	VARCHAR(255)	NULL	Client computer name
TERMINAL	VARCHAR(255)	NULL	Identifier for the user's terminal
TIMESTAMP	DATE	NULL	Date and time of creation of the audit trail entry (in the local database session time zone)
OWNER	VARCHAR(128)	NULL	Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR(128)	NULL	Name of the object affected by the action. Expected values are: • ROLE$ • REALM$ • CODE$ • FACTOR$
ACTION	NUMBER	NOT NULL	Numeric action type code. The corresponding name of the action type is in the ACTION_NAME column. See Table 24-4 for a listing of the possible actions.
ACTION_NAME	VARCHAR(128)	NULL	Name of the action type corresponding to the numeric code in the ACTION column
ACTION_OBJECT_ID	NUMBER	NULL	The unique identifier of the record in the table specified under OBJ_NAME
ACTION_OBJECT_NAME	VARCHAR(128)	NULL	The unique name or natural key of the record in the table specified under OBJ_NAME
ACTION_COMMAND	VARCHAR2(4000)	NULL	The SQL text of the command procedure that was run that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)	NULL	The labels for all (traditional) 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. Starting with Oracle Database 23c, traditional auditing is desupported.
RULE_SET_ID	NUMBER	NULL	The unique identifier of the rule set that was executing and caused the audit event to trigger

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NULL	The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER	NULL	Not used
RULE_NAME	VARCHAR2(128)	NULL	Not used
FACTOR_CONTEXT	VARCHAR2(4000)	NULL	An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered
COMMENT_TEXT	VARCHAR2(4000)	NULL	Text comment on the audit trail entry, providing more information about the statement audited
SESSIONID	NUMBER	NOT NULL	Numeric identifier for each Oracle session
ENTRYID	NUMBER	NOT NULL	Same as the value in the ID# column
STATEMENTID	NUMBER	NOT NULL	Numeric identifier for the statement invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
RETURNCODE	NUMBER	NOT NULL	Oracle error code generated by the action. The error code for a statement or procedure invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
EXTENDED_TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	NULL	Time stamp of creation of the audit trail entry (time stamp of user login for entries) in UTC (Coordinated Universal Time) time zone
PROXY_SESSIONID	NUMBER	NULL	Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)	NULL	Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER	NULL	Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)	NULL	Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(128)	NULL	Database login user name of the user whose actions were audited
CREATE_DATE	DATE	NULL	Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(128)	NULL	Same as CREATED_BY column value
UPDATE_DATE	DATE	NULL	Same as UPDATED_BY column value

The following table describes the possible values for the ACTION column of the DVSYS.DV$ENFORCEMENT_AUDIT view.

Table 24-4 DVSYS.DV$ENFORCEMENT_AUDIT View ACTION Values

Action Type Code	Action Name
10000	Factor Evaluation Audit
10001	Factor Assignment Audit
10002	Factor Expression Audit
10003	Realm Violation Audit
10004	Realm Authorization Audit
10005	Command Authorization Audit
10006	Secure Role Audit
10007	Session Initialization Audit
10008	Secure Command Authorization Audit
10009	OLS Session Initialization Audit
10010	OLS Attempt to Upgrade Label Audit
10011	Command Failure Audit

Related Topics

- AUDSYS.DV$ENFORCEMENT_AUDIT View
  The `AUDSYS.DV$ENFORCEMENT_AUDIT` view is almost the same as the `DVSYS.DV$ENFORCEMENT_AUDIT` view except that it captures unified audit trail Database Vault audit records.

24.46 DVSYS.DV$REALM View

The `DVSYS.DV$REALM` data dictionary view describes settings that were used to create Oracle Database Vault realms, such as which audit options have been assigned or whether the realm is a mandatory realm.

This view also indicates information such as who created and updated the realm, and when the realm was created and updated.

For example:

```
SELECT NAME, CREATED_BY, TYPE FROM DVSYS.DV$REALM WHERE NAME LIKE 'Statistics';
```

Output similar to the following appears:

```
NAME                          CREATED_BY TYPE
----------------------------- ---------- -----
Performance Statistics Realm JGODFREY   2
```

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	ID number of the realm
NAME	VARCHAR2(128)	NOT NULL	Name of the realm
DESCRIPTION	VARCHAR2(1024)	NULL	Description of the realm

Column	Datatype	Null	Description
AUDIT_OPTIONS	NUMBER	NOT NULL	Audit options set for the realm. See audit_options in the UPDATE_REALM procedure description. See Related Topics.
REALM_TYPE	NUMBER	NULL	Type of realm: whether it is a regular realm or a mandatory realm. See realm_type in the UPDATE_REALM procedure description. See Related Topics.
COMMON	VARCHAR2(3)	NULL	Indicates whether the realm is local or common. Possible values are: • YES if the realm is common • NO if the realm is local
INHERITED	VARCHAR2(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ENABLED	VARCHAR2(1)	NOT NULL	Whether the realm has been enabled. See enabled in the DBMS_MACADM.UPDATE_REALM procedure description. See Related Topics.
VERSION	NUMBER	NULL	Version of Oracle Database Vault in which the realm was created
CREATED_BY	VARCHAR2(128)	NULL	User who created the realm
CREATE_DATE	DATE	NULL	Date on which the realm was created.
UPDATED_BY	VARCHAR2(128)	NULL	User who last updated the realm
UPDATE_DATE	DATE	NULL	Date on which the realm was last updated

Related Topics

- DBA_DV_REALM View
  The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.
- UPDATE_REALM Procedure
  The UPDATE_REALM procedure updates a realm.

24.47 DVSYS.DBA_DV_COMMON_OPERATION_STATUS View

The DVSYS.DBA_DV_COMMON_OPERATION_STATUS data dictionary view displays the status of the DBMS_MACADM.ALLOW_COMMON_OPERATION procedure setting.

For example:

```
SELECT * FROM DVSYS.DBA_DV_COMMON_OPERATION_STATUS;
```

Output similar to the following appears:

```
NAME                        STATUS
--------------------------  ------
DV_ALLOW_COMMON_OPERATION   FALSE
```

Column	Datatype	Null	Description
NAME	CHAR(25)	NOT NULL	Name of this control, that is, DV_ALLOW_COMMON_OPERATION
STATUS	VARCHAR(5)	NOT NULL	Either of the following: • TRUE prevents local users from creating Oracle Database Vault controls on common user objects. This setting applies to existing local PDB Database Vault controls that were created on common user objects, so that they will not be enforced on common users. • FALSE enables local users to create Database Vault controls on common user objects. Existing local PDB controls that were created on common user objects will continue to be enforced.

24.48 DVSYS.POLICY_OWNER_COMMAND_RULE View

The DVSYS.POLICY_OWNER_COMMAND_RULE data dictionary view enables DV_POLICY_OWNER role users to find information about the command rules that are used by Database Vault policies.

Examples of information that users can find include the command rule name, its associated rule set, and whether it is enabled. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT COMMAND, OBJECT_OWNER, OBJECT_NAME FROM DVSYS.POLICY_OWNER_COMMAND_RULE;
```

Output similar to the following appears:

```
COMMAND       OBJECT_OWNER  OBJECT_NAME
------------- ------------- ------------
SELECT        HR            EMPLOYEES
```

Column	Datatype	Null	Description
COMMAND	VARCHAR(128)	NOT NULL	Name of the command rule.
CLAUSE_NAME	VARCHAR(100)	NOT NULL	A clause from either the ALTER SYSTEM or ALTER SESSION SQL statement, which was used to create the command rule. For example, you it could list the SET clause for the ALTER SESSION statement. The command rule settings for these two statements are described in the DBMS_MACADM.CREATE_COMMAND_RULE procedure. See Related Topics.
PARAMETER_NAME	VARCHAR(128)	NOT NULL	A parameter from the ALTER SYSTEM or ALTER SESSION command rule CLAUSE_NAME setting. See Related Topics.
EVENT_NAME	VARCHAR(128)	NOT NULL	An event that the ALTER SYSTEM or ALTER SESSION command rule defines. See Related Topics.
COMPONENT_NAME	VARCHAR(128)	NOT NULL	A component of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule. See Related Topics.
ACTION_NAME	VARCHAR(128)	NOT NULL	An action of the EVENT_NAME setting for the ALTER SYSTEM or ALTER SESSION command rule. See Related Topics.
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set associated with this command rule.
OBJECT_OWNER	VARCHAR(128)	NOT NULL	The owner of the object that the command rule affects.
OBJECT_NAME	VARCHAR(128)	NOT NULL	The name of the database object the command rule affects (for example, a database table).
ENABLED	VARCHAR(1)	NOT NULL	Y indicates the command rule is enabled; N indicates it is disabled.
PRIVILEGE_SCOPE	NUMBER	NOT NULL	Obsolete column
ID#	NUMBER	NOT NULL	The ID number of the command rule, which is automatically generated when the command rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the command rule is a default (that is, Oracle-supplied) command rule or a user-created command rule. Possible values are: • YES if the command rule is a default command rule • NO if the command rule is a user-created command rule

Related Topics

- CREATE_COMMAND_RULE Procedure
  The CREATE_COMMAND_RULE procedure creates both command and local command rules, which can be added to a rule set.
- DVSYS.POLICY_OWNER_POLICY View
  The DVSYS.POLICY_OWNER_POLICY data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners.

24.49 DVSYS.POLICY_OWNER_POLICY View

The DVSYS.POLICY_OWNER_POLICY data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information such as the names, descriptions, and states of existing policies in the current database instance, including policies created by other policy owners.

The columns of the DVSYS.POLICY_OWNER_POLICY view are the same as those in DBA_DV_POLICY. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT POLICY_NAME, STATE FROM DVSYS.POLICY_OWNER_POLICY
  WHERE STATE != 'ENABLED';
```

Output similar to the following appears:

```
POLICY_NAME                       STATE
--------------------------------- --------
HR.EMPLOYEES_pol                  ENABLED
```

Related Topics

- DBA_DV_POLICY View
  The DBA_DV_POLICY data dictionary view lists the Oracle Database Vault policies that were created in the current database instance.

24.50 DVSYS.POLICY_OWNER_REALM View

The POLICY_OWNER_REALM data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the realms that have been associated with Database Vault policies.

Examples of information that users can find include the realm name, audit options, type, whether it is inherited, and if it is enabled. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT NAME, ENABLED FROM DVSYS.POLICY_OWNER_REALM;
```

Output similar to the following appears:

```
NAME                      ENABLED
------------------------- --------
HR.EMPLOYEES_realm        S
```

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Names of the realms that have been associated with Database Vault policies.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the realm

Column	Datatype	Null	Description
AUDIT_OPTIONS	NUMBER	NOT NULL	Audit options using traditional auditing set for the realm. See audit_options in the UPDATE_REALM command description. See Related Topics for a description of the possible values. Starting with Oracle Database 23c, traditional auditing is desupported.
REALM_TYPE	NUMBER	NULL	Type of realm: whether it is a regular realm or a mandatory realm. See realm_type in the UPDATE_REALM command description. See Related Topics.
COMMON_REALM	VARCHAR2(3)	NULL	Indicates whether the realm is local or common. Possible values are: • YES if the realm is common • NO if the realm is local
INHERITED_REALM	VARCHAR2(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ENABLED	VARCHAR2(1)	NOT NULL	Indicates the enablement status of the realm. Possible values are: • Y for yes (enabled) • N for no (not enabled) • S for simulation mode
ID#	NUMBER	NOT NULL	The ID number of the realm, which is automatically generated when the realm is created
ORACLE_SUPPLIED	VARCHAR(3)	NOT NULL	Indicates whether the realm is a default (that is, Oracle-supplied) realm or a user-created realm. Possible values are: • YES if the realm is a default realm • NO if the realm is a user-created realm

Related Topics

- DVSYS.POLICY_OWNER_REALM_AUTH View
  The DVSYS.POLICY_OWNER_REALM_AUTH data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the authorization that was granted to realms that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_REALM_OBJECT View
  The DVSYS.POLICY_OWNER_REALM_OBJECT data dictionary view enables users to find information about the objects that have been added to realms that are associated

with Database Vault policies, such as. Only users who have been granted the DV_POLICY_OWNER role can query this view.

- UPDATE_REALM Procedure
  The UPDATE_REALM procedure updates a realm.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

24.51 DVSYS.POLICY_OWNER_REALM_AUTH View

The DVSYS.POLICY_OWNER_REALM_AUTH data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the authorization that was granted to realms that have been associated with Database Vault policies.

Examples of the information that users can find are the realm name, grantee, and associated rule set. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT REALM_NAME, INHERITED_REALM FROM DVSYS.POLICY_OWNER_REALM_AUTH;
```

Output similar to the following appears:

```
REALM_NAME                INHERITED
------------------------- --------
HR.EMPLOYEES_realm        NO
```

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(128)	NOT NULL	Names of the realms that have been associated with Database Vault policies. See also Related Topics.
COMMON_REALM	VARCHAR2(3)	NULL	Indicates whether the realm is local or common.
INHERITED_REALM	VARCHAR2(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
GRANTEE	VARCHAR(128)	NOT NULL	User or role name to authorize as owner or participant.
AUTH_RULE_SET_NAME	VARCHAR(128)	NULL	Rule set to check before authorizing. If the rule set evaluates to TRUE, then the authorization is allowed.
AUTH_OPTIONS	VARCHAR(4000)	NULL	Type of realm authorization: either Participant or Owner.

Column	Datatype	Null	Description
COMMON_AUTH	VARCHAR(3)	NULL	Indicates whether the user who is authorized for this realm is local or common. Possible values are: • YES if the user is a common user • NO if the users is a local user
INHERITED_AUTH	VARCHAR(3)	NULL	Possible values are: • YES • NO

Related Topics

- DBA_DV_REALM View
  The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.
- DVSYS.POLICY_OWNER_REALM View
  The POLICY_OWNER_REALM data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the realms that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_REALM_OBJECT View
  The DVSYS.POLICY_OWNER_REALM_OBJECT data dictionary view enables users to find information about the objects that have been added to realms that are associated with Database Vault policies, such as. Only users who have been granted the DV_POLICY_OWNER role can query this view.

24.52 DVSYS.POLICY_OWNER_REALM_OBJECT View

The DVSYS.POLICY_OWNER_REALM_OBJECT data dictionary view enables users to find information about the objects that have been added to realms that are associated with Database Vault policies, such as. Only users who have been granted the DV_POLICY_OWNER role can query this view.

Examples of information that users can find include the realm name, grantee, and associated rule set.

For example:

```
SELECT REALM_NAME, OWNER, OBJECT_NAME, OBJECT_TYPE FROM
DVSYS.POLICY_OWNER_REALM_OBJECT;
```

Output similar to the following appears:

```
REALM_NAME          OWNER  OBJECT_NAME OBJECT_TYPE
------------------- ------ ----------- -----------
HR.EMPLOYEES_realm HR     EMPLOYEES   TABLE
```

Column	Datatype	Null	Description
REALM_NAME	VARCHAR(128)	NOT NULL	Names of the realms that have been associated with Database Vault policies. See also Related Topics.
COMMON_REALM	VARCHAR2(3)	NULL	Indicates whether the realm is local or common.

Column	Datatype	Null	Description
INHERITED_REALM	VARCHAR2(3)	NULL	Shows the inheritance status of the realm, when the COMMON column output is YES. Values are as follows: • YES means that the realm was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the realm is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
OWNER	VARCHAR(128)	NOT NULL	Database schema owner who owns the object.
OBJECT_NAME	VARCHAR(128)	NOT NULL	Name of the object the realm protects.
OBJECT_TYPE	VARCHAR(32)	NOT NULL	Type of object the realm protects, such as a database table, view, index, or role.

Related Topics

- DBA_DV_REALM View
  The DBA_DV_REALM data dictionary view lists the realms created in the current database instance.
- DVSYS.POLICY_OWNER_REALM View
  The POLICY_OWNER_REALM data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the realms that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_REALM_AUTH View
  The DVSYS.POLICY_OWNER_REALM_AUTH data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the authorization that was granted to realms that have been associated with Database Vault policies.

24.53 DVSYS.POLICY_OWNER_RULE View

The DVSYS.POLICY_OWNER_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rules that have been associated with rule sets in Database Vault policies, such as the rule name and its expression. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT NAME, RULE_EXPR FROM DVSYS.POLICY_OWNER_RULE WHERE NAME = 'True';
```

Output similar to the following appears:

```
NAME       RULE_EXPR
---------- --------
True       1=1
```

Column	Datatype	Null	Description
NAME	VARCHAR(128)	NOT NULL	Name of the rule.
RULE_EXPR	VARCHAR(1024)	NOT NULL	PL/SQL expression for the rule.
COMMON	VARCHAR(3)	NOT NULL	Indicates whether the rule is local or common. Possible values are: • YES if the rule is common • NO if the rule is local
INHERITED	VARCHAR(3)	NULL	Shows the inheritance status of the rule, when the COMMON column output is YES. Values are as follows: • YES means that the rule was defined in another container that is higher in the hierarchy of the container tree, and inherited in this container when the Database Vault policy was synced during the synchronization process of applications in an application PDB. • NO means that the rule is a local object, or it is common from that container. For example, in an application root, an application common realm will have an INHERITED value NO but a CDB root common command rule will have an INHERITED value of YES.
ID#	NUMBER	NOT NULL	The ID number of the rule, which is automatically generated when the rule is created
ORACLE_SUPPLIED	VARCHAR(3)	NULL	Indicates whether the rule is a default (that is, Oracle-supplied) rule or a user-created rule. Possible values are: • YES if the rule is a default rule • NO if the rule is a user-created rule

Related Topics

- DVSYS.POLICY_OWNER_COMMAND_RULE View
  The DVSYS.POLICY_OWNER_COMMAND_RULE data dictionary view enables DV_POLICY_OWNER role users to find information about the command rules that are used by Database Vault policies.
- DVSYS.POLICY_OWNER_RULE_SET View
  The DVSYS.POLICY_OWNER_RULE_SET data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies.

24.54 DVSYS.POLICY_OWNER_RULE_SET View

The DVSYS.POLICY_OWNER_RULE_SET data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies.

Examples of information that users can find include the rule set name, its handler information, and whether it is enabled. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT RULE_SET_NAME, ENABLED FROM DVSYS.POLICY_OWNER_RULE_SET;
```

Output similar to the following appears:

```
RULE_SET_NAME  ENABLED
-------------- --------
Allow Sessions Y
```

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set.
DESCRIPTION	VARCHAR(1024)	NULL	Description of the rule set.
ENABLED	VARCHAR(1)	NOT NULL	Indicates whether the rule set has been enabled. Y (Yes) enables the rule set; N (No) disables it.
EVAL_OPTIONS_MEANING	VARCHAR(4000)	NULL	For rules sets that contain multiple rules, determines how many rules are evaluated. Possible values are: • All True: All rules in the rule set must evaluate to true for the rule set itself to evaluate to TRUE. • Any True: At least one rule in the rule set must evaluate to true for the rule set itself to evaluate to TRUE.
AUDIT_OPTIONS	NUMBER	NOT NULL	Indicates when auditing using traditional auditing is used. Possible values are: • 0: No auditing • 1: Audit on failure • 2: Audit on success • 3: Audit on both failure and success Starting with Oracle Database 23c, traditional auditing is desupported.
FAIL_OPTIONS_MEANING	VARCHAR(4000)	NULL	Determines when an audit record is created for the rule set. Possible values are: • Do Not Show Error Message. • Show Error Message
FAIL_MESSAGE	VARCHAR(80)	NULL	Error message for failure that is associated with the fail code listed in the FAIL_CODE column.
FAIL_CODE	VARCHAR(10)	NULL	The error message number associated with the message listed in the FAIL_MESSAGE column. Possible values are in the ranges of -20000 to -20999 or 20000 to 20999.
HANDLER_OPTIONS	NUMBER	NOT NULL	Determines how error handling is used. Possible values are: • 0: Disables error handling. • 1: Call handler on rule set failure. • 2: Call handler on rule set success.
HANDLER	VARCHAR(1024)	NULL	Name of the PL/SQL function or procedure that defines the custom event handler logic.
IS_STATIC	VARCHAR2(5)	NULL	Indicates how often the rule set is evaluated during a user session. Possible values are: • TRUE: The rule set is evaluated once, and result of the rule set is reused throughout the user session. • FALSE (default): The rule set is evaluated each time it is accessed during the user session.
ID#	NUMBER)	NOT NULL	The ID number of the rule set, which is automatically generated when the rule set is created

Column	Datatype	Null	Description
ORACLE_SUPPLIED	VARCHAR2(3)	NULL	Indicates whether the rule set is a default (that is, Oracle-supplied) rule set or a user-created rule set. Possible values are: • YES if the rule set is a default rule set • NO if the rule set is a user-created rule set

Related Topics

- DVSYS.POLICY_OWNER_COMMAND_RULE View
  The DVSYS.POLICY_OWNER_COMMAND_RULE data dictionary view enables DV_POLICY_OWNER role users to find information about the command rules that are used by Database Vault policies.
- DVSYS.POLICY_OWNER_RULE View
  The DVSYS.POLICY_OWNER_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rules that have been associated with rule sets in Database Vault policies, such as the rule name and its expression. Only users who have been granted the DV_POLICY_OWNER role can query this view.
- DVSYS.POLICY_OWNER_RULE_SET View
  The DVSYS.POLICY_OWNER_RULE_SET data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies.
- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

24.55 DVSYS.POLICY_OWNER_RULE_SET_RULE View

The DVSYS.POLICY_OWNER_RULE_SET_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that contain rules used in Database Vault policies.

Examples of information that users can find include the rule set name and whether it is enabled. Only users who have been granted the DV_POLICY_OWNER role can query this view.

For example:

```
SELECT ENABLED FROM DVSYS.POLICY_OWNER_RULE_SET_RULE WHERE RULE_SET_NAME = 'Can
Maintain Own Account';
```

Output similar to the following appears:

```
ENABLED
----------
Y
```

Column	Datatype	Null	Description
RULE_SET_NAME	VARCHAR(128)	NOT NULL	Name of the rule set that contains the rule.
RULE_NAME	VARCHAR(128)	NOT NULL	Name of the rule.

Column	Datatype	Null	Description
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.

Related Topics

- DVSYS.POLICY_OWNER_COMMAND_RULE View
  The DVSYS.POLICY_OWNER_COMMAND_RULE data dictionary view enables DV_POLICY_OWNER role users to find information about the command rules that are used by Database Vault policies.
- DVSYS.POLICY_OWNER_RULE_SET View
  The DVSYS.POLICY_OWNER_RULE_SET data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rule sets that have been associated with Database Vault policies.
- DVSYS.POLICY_OWNER_RULE View
  The DVSYS.POLICY_OWNER_RULE data dictionary view enables users who have been granted the DV_POLICY_OWNER role to find information about the rules that have been associated with rule sets in Database Vault policies, such as the rule name and its expression. Only users who have been granted the DV_POLICY_OWNER role can query this view.

24.56 AUDSYS.DV$CONFIGURATION_AUDIT View

The AUDSYS.DV$CONFIGURATION_AUDIT view is almost the same as the DVSYS.DV$CONFIGURATION_AUDIT view except that it captures unified audit trail Database Vault audit records.

Related Topics

- DVSYS.DV$CONFIGURATION_AUDIT View
  The DVSYS.DV$CONFIGURATION_AUDIT data dictionary view captures DVSYS.AUDIT_TRAIL$ table audit trail records.

24.57 AUDSYS.DV$ENFORCEMENT_AUDIT View

The AUDSYS.DV$ENFORCEMENT_AUDIT view is almost the same as the DVSYS.DV$ENFORCEMENT_AUDIT view except that it captures unified audit trail Database Vault audit records.

Related Topics

- DVSYS.DV$ENFORCEMENT_AUDIT View
  The DVSYS.DV$ENFORCEMENT_AUDIT data dictionary view provides information about enforcement-related audits from the DVSYS.AUDIT_TRAIL$ table.

25
Monitoring Oracle Database Vault

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

- About Monitoring Oracle Database Vault
  You can use the Database Vault home page in Oracle Enterprise Manager Cloud Control to monitor a Database Vault-enabled database.
- Monitoring Security Violations and Configuration Changes
  A user who has been granted the appropriate role can use Oracle Database Vault Administrator to monitor security violations and configuration changes.

25.1 About Monitoring Oracle Database Vault

You can use the Database Vault home page in Oracle Enterprise Manager Cloud Control to monitor a Database Vault-enabled database.

This feature displays the top five attempted violations and who the top five attempted violators are. The attempted violations cover violations to realms and to command rules. The attempted violators is categorized into users and client hosts. By clicking the **Oracle Database Vault** link under Top 5 Attempted Violations, you can find details such as the type of violation, when it occurred, who the user was, and so on. Similarly, if you click the user link (for example, **SYS**) under Top 5 Attempted Violators, you can find detailed information about each violator, such as the action they performed, the client host name where the action originated, and when the violation occurred. You can manually refresh the data, and restrict the data view, such as within the last 24 hours. This page also shows a table listing all alerts that have been generated.

Before you can view these events, if you have not migrated your database to unified auditing, then you must ensure that the `AUDIT_TRAIL` initialization parameter (deprecated) is set to `DB` or `DB, EXTENDED`. If you have migrated your database to use unified auditing, then you do not need to configure any additional settings. You are ready to check for security violations.

Related Topics

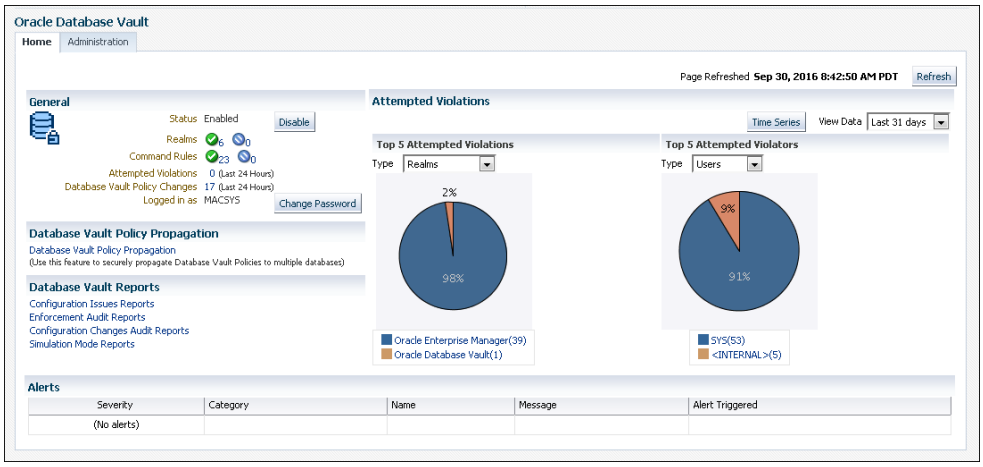
- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.

25.2 Monitoring Security Violations and Configuration Changes

A user who has been granted the appropriate role can use Oracle Database Vault Administrator to monitor security violations and configuration changes.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. Select the **Home** tab.

A page similar to the following appears:



3. To find attempted violations for a specific time, such as the last 7 days, select from the menu under the **Time Series** button in the upper right corner.

   You also can change the pie chart to a graph by clicking the **Time Series** button.

4. To find the **Configuration Issues Reports**, **Enforcement Audit Reports**, **Configuration Changes Audit Reports**, and **Simulation Mode Reports**, select the appropriate link under Database Vault reports.

   See Oracle Database Vault Reports for detailed information about the Database Vault reports.

26
Oracle Database Vault Reports

Oracle Enterprise Manager provides Oracle Database Vault-related reports.

- About the Oracle Database Vault Reports
  Oracle Database Vault provides reports that display security-related information from the database.
- Who Can Run the Oracle Database Vault Reports?
  Users must have the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role before they can run the Oracle Database Vault reports.
- Running the Oracle Database Vault Reports
  A user who has been granted the appropriate roles can run the Oracle Database Vault reports from Database Vault Administrator.
- Oracle Database Vault Configuration Issues Reports
  The configuration issues reports track the settings for command rules, rule sets, realms, and other Oracle Database Vault configurations.
- Oracle Database Vault Auditing Reports
  The Oracle Database Vault audit reports capture the results of unified audit policies.
- Oracle Database Vault General Security Reports
  The general security reports track information such as object privileges related to `PUBLIC` or privileges granted to a database account or role.

26.1 About the Oracle Database Vault Reports

Oracle Database Vault provides reports that display security-related information from the database.

These reports also show custom Oracle Database Vault audit event information. If you have unified auditing enabled, then the reports capture the results of your unified audit policies.

The reports are in two categories:

- **Database Vault Reports.** These reports allow you to check configuration issues with realms, command rules, factors, factor identities, rule sets, and secure application roles. These reports also reveal realm violations, auditing results, and so on.
- **General Security Reports.** These reports allow you to check the status of object privileges, database account system privileges, sensitive objects, privilege management, powerful database accounts and roles, initialization parameters, profiles, account passwords, security audits, and other security vulnerability reports.

Related Topics

- Oracle Database Vault-Specific Reports in Enterprise Manager Cloud Control
  From the Database Vault home page, you can find information about violations.
- Oracle Database Vault Data Dictionary Views
  You can find information about the Oracle Database Vault configuration settings by querying the Database Vault-specific data dictionary views.

26.2 Who Can Run the Oracle Database Vault Reports?

Users must have the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role before they can run the Oracle Database Vault reports.

Related Topics

- DV_OWNER Database Vault Owner Role
  The `DV_OWNER` role enables you to manage the Oracle Database Vault roles and its configuration.
- DV_ADMIN Database Vault Configuration Administrator Role
  The `DV_ADMIN` role controls the Oracle Database Vault PL/SQL packages.
- DV_SECANALYST Database Vault Security Analyst Role
  The `DV_SECANALYST` role enables users to analyze activities.

26.3 Running the Oracle Database Vault Reports

A user who has been granted the appropriate roles can run the Oracle Database Vault reports from Database Vault Administrator.

1. Log in to Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER`, `DV_ADMIN`, or `DV_SECANALYST` role and the `SELECT ANY DICTIONARY` privilege. Logging in to Oracle Database Vault from Oracle Enterprise Cloud Control explains how to log in.
2. In the Home page, under Reports, select **Database Vault Reports**.
3. On the left side, select the category of reports that you want.
   - Database Vault Configuration Issues
   - Database Vault Enforcement Audit Reports
   - Database Vault Configuration Changes
4. In the Reports page, expand the category that contains the report.

   For example, to find the Rule Set Configurations Issues report, you must expand **Database Vault Configuration Issues**.
5. Select the report (for example, **Rule Set Configuration Issues**).

   The report appears in the right pane.
6. Optionally, use the **Search** field to filter the report.

   For example, you can search for reported incidents that involve a specific rule set. The Search field contents vary depending on the report.
7. When you finished viewing the report, click the **OK** button.

26.4 Oracle Database Vault Configuration Issues Reports

The configuration issues reports track the settings for command rules, rule sets, realms, and other Oracle Database Vault configurations.

- Command Rule Configuration Issues Report
  The Command Rule Configuration Issues Report displays command rules that have configuration issues.
- Rule Set Configuration Issues Report
  The Rule Set Configuration Issues Report displays Oracle Database Vault rule set configuration issues.
- Realm Authorization Configuration Issues Report
  The Realm Authorization Configuration Issues Report displays Oracle Database Vault realm configuration issues.
- Factor Configuration Issues Report
  The Factor Configuration Issues Report displays Oracle Database Vault factors configuration issues.
- Factor Without Identities Report
  The Factor Without Identities Report displays Oracle Database Vault factors that have no identities configured.
- Identity Configuration Issues Report
  The Identity Configuration Issues Report displays Oracle Database Vault factor identity configuration issues.
- Secure Application Configuration Issues Report
  The Secure Application Configuration Issues Report displays Database Vault secure application role configuration issues.

26.4.1 Command Rule Configuration Issues Report

The Command Rule Configuration Issues Report displays command rules that have configuration issues.

These issues are as follows:

- Rule set for the command rule is disabled.
- Rule set for the command rule is incomplete.
- Object owner for the command rule does not exist. This can happen when the user account for the object has been dropped.

26.4.2 Rule Set Configuration Issues Report

The Rule Set Configuration Issues Report displays Oracle Database Vault rule set configuration issues.

This report tracks when no rules are defined or enabled for a rule set.

26.4.3 Realm Authorization Configuration Issues Report

The Realm Authorization Configuration Issues Report displays Oracle Database Vault realm configuration issues.

These issues are as follows:

- Rule set for a realm authorization is disabled.
- Grantee does not exist for a realm authorization.

- Owner does not exist for a realm-secured object. This can happen when the user account has been dropped.

In most cases, however, these types of issues are caught when you configure the realm and during validation.

26.4.4 Factor Configuration Issues Report

The Factor Configuration Issues Report displays Oracle Database Vault factors configuration issues.

These issues are as follows:

- Rule set for factor assignment is disabled.
- Rule set for factor assignment is incomplete.
- Audit options for the factor are invalid.
- No factor retrieval method or constant exists.
- No subfactors (that is, child factors) are linked to a factor identity.
- No subfactors (child factors) are linked to a label factor.
- Oracle Label Security policy does not exist for the factor.

26.4.5 Factor Without Identities Report

The Factor Without Identities Report displays Oracle Database Vault factors that have no identities configured.

For some factors such as `Background_Job_Id`, this may not be a real problem, but the report can help you determine whether your access control configuration is complete and whether you have accounted for all factor configuration.

26.4.6 Identity Configuration Issues Report

The Identity Configuration Issues Report displays Oracle Database Vault factor identity configuration issues.

These issues are as follows:

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

26.4.7 Secure Application Configuration Issues Report

The Secure Application Configuration Issues Report displays Database Vault secure application role configuration issues.

These issues are as follows:

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

26.5 Oracle Database Vault Auditing Reports

The Oracle Database Vault audit reports capture the results of unified audit policies.

- Realm Audit Report
  The Realm Audit Report shows audit records generated by the realm protection and realm authorization operations.
- Command Rule Audit Report
  The Command Rule Audit Report shows audit records generated by command rule processing operations.
- Factor Audit Report
  The Factor Audit Report shows factors that failed to evaluate or were set to create audit records under various conditions.
- Label Security Integration Audit Report
  The Label Security Integration Audit Report shows audit records the session initialization operation generates and the session label assignment operation of label security.
- Core Database Vault Audit Trail Report
  The Core Database Vault Audit Trail Report shows audit records that the core access security session initialization operation generates.
- Secure Application Role Audit Report
  The Secure Application Role Audit Report shows the audit records that the Oracle Database Vault secure application role-enabling operation generates.

26.5.1 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.

26.5.2 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.

26.5.3 Factor Audit Report

The Factor Audit Report shows factors that failed to evaluate or were set to create audit records under various conditions.

This report also shows failed attempts to set factors.

You can audit instances where a factor identity cannot be resolved and assigned (such as *No data found* or *Too many rows*). A factor can have an associated rule set that assigns an

identity to the factor at run time. When you configure a factor, you can set it to audit the rule set processing results.

26.5.4 Label Security Integration Audit Report

The Label Security Integration Audit Report shows audit records the session initialization operation generates and the session label assignment operation of label security.

You can audit instances where the label security session fails to initialize, and where the label security component prevents a session from setting a label that exceeds the maximum session label.

26.5.5 Core Database Vault Audit Trail Report

The Core Database Vault Audit Trail Report shows audit records that the core access security session initialization operation generates.

You can audit instances where the access security session fails to initialize. It displays the following data:

Data A-R	Data R-U
Account	Rule Set
Command	Timestamp
Instance Number	Rule Set
Object Name	User Host
Return Code	-

26.5.6 Secure Application Role Audit Report

The Secure Application Role Audit Report shows the audit records that the Oracle Database Vault secure application role-enabling operation generates.

Related Topics

- Configuring Secure Application Roles for Oracle Database Vault
  Secure application roles enable you to control how much access users have to an application.

26.6 Oracle Database Vault General Security Reports

The general security reports track information such as object privileges related to `PUBLIC` or privileges granted to a database account or role.

- Object Privilege Reports
  The object privilege reports track privileges affected by `PUBLIC`, direct object privileges, and object dependencies.
- Database Account System Privileges Reports
  The database account system privileges reports track activities such as direct, indirect, hierarchical, and `ANY` system privileges.

- Sensitive Objects Reports
  The sensitive objects reports track activities such as grants on the `EXECUTE` privilege on `SYS` schema objects and access to sensitive objects.
- Privilege Management - Summary Reports
  The privilege management summary reports track privilege distribution by grantees, owners, and privileges.
- Powerful Database Accounts and Roles Reports
  The powerful database accounts and roles reports track information about users who have been granted power privileges, such as the `WITH ADMIN` privilege.
- Initialization Parameters and Profiles Reports
  The initialization parameters and profiles reports track database parameters, resource profiles, and system limits.
- Database Account Password Reports
  The database account password reports track default passwords and account statuses of database accounts.
- Security Audit Report: Core Database Audit Report
  The Core Database Audit Report lists database audit trail records..
- Other Security Vulnerability Reports
  Other security vulnerability reports track vulnerabilities that arise with activities such as Java policy grants ir operating system directory objects.

26.6.1 Object Privilege Reports

The object privilege reports track privileges affected by `PUBLIC`, direct object privileges, and object dependencies.

- Object Access By PUBLIC Report
  The Object Access By PUBLIC Report lists all objects whose access has been granted to `PUBLIC`.
- Object Access Not By PUBLIC Report
  The Object Access Not By PUBLIC Report describes the object access used by the database accounts on the Report Parameters page.
- Direct Object Privileges Report
  The Direct Object Privileges Report shows the direct object privileges granted to *nonsystem* database accounts.
- Object Dependencies Report
  The Object Dependencies Report describes dependencies in the database between procedures, packages, functions, package bodies, and triggers.

26.6.1.1 Object Access By PUBLIC Report

The Object Access By PUBLIC Report lists all objects whose access has been granted to `PUBLIC`.

This report details all the object access the database accounts that you specify on the Report Parameters page, through object grants to `PUBLIC`. On the Reports Parameters page, you can filter the results based on the privilege, the object owner, or the object name.

> **Note:**
>
> This report can be quite large if you choose the defaults.

26.6.1.2 Object Access Not By PUBLIC Report

The Object Access Not By PUBLIC Report describes the object access used by the database accounts on the Report Parameters page.

It checks the grants to the account directly or through a role, but excluding the grants to `PUBLIC`.

On the Reports Parameters page, you can filter the results based on the privilege, the object owner or the object name.

> **Note:**
>
> This report can be quite large if you choose the defaults.

26.6.1.3 Direct Object Privileges Report

The Direct Object Privileges Report shows the direct object privileges granted to *nonsystem* database accounts.

The following database accounts are excluded from the report:

Accounts C-O	Accounts P-W
CTXSYS	PUBLIC
DMSYS	SYS
DVSYS	SYSMAN
LBACSYS	SYSTEM
MDSYS	WKSYS
ORDSYS	WMSYS

26.6.1.4 Object Dependencies Report

The Object Dependencies Report describes dependencies in the database between procedures, packages, functions, package bodies, and triggers.

The report includes dependencies on views created without any database links.

This report can help you develop a security policy using the principle of least privilege for existing applications. If a database object, such as a `UTL_FILE` package, has privileges granted to `PUBLIC` or some other global role, then you can use the Object Dependencies Report to determine an account that may depend on the object and to determine how the account uses the object. To run the report, enter the database

account you are inspecting for dependency and the object it may be dependent on, in the Report Parameters page.

The Report Results page shows the dependent object and object type and the source object name and type. This report shows where the potentially sensitive object is being used. By looking at several accounts, you might be able to see patterns that can help you develop restricted roles. These restricted roles can replace `PUBLIC` grants on widely used sensitive objects.

26.6.2 Database Account System Privileges Reports

The database account system privileges reports track activities such as direct, indirect, hierarchical, and `ANY` system privileges.

- Direct System Privileges By Database Account Report
  The Direct System Privileges By Database Account Report lists system privileges directly granted to the database account selected on the Report Parameters page.
- Direct and Indirect System Privileges By Database Account Report
  The Direct and Indirect System Privileges By Database Account Report displays system privileges for the database account selected on the Report Parameters page.
- Hierarchical System Privileges by Database Account Report
  The Hierarchical System Privileges by Database Account Report shows a hierarchical breakdown of role-based system privileges and direct system privileges.
- ANY System Privileges for Database Accounts Report
  The ANY System Privileges for Database Accounts Report shows `ANY` system privileges granted to the specified database account or role.
- System Privileges By Privilege Report
  The System Privileges By Privilege Report lists database accounts and roles that have the system privilege selected on the Report Parameters page.

26.6.2.1 Direct System Privileges By Database Account Report

The Direct System Privileges By Database Account Report lists system privileges directly granted to the database account selected on the Report Parameters page.

This report also shows whether a privilege has been granted the `WITH ADMIN` option.

26.6.2.2 Direct and Indirect System Privileges By Database Account Report

The Direct and Indirect System Privileges By Database Account Report displays system privileges for the database account selected on the Report Parameters page.

The system privileges may have been granted directly or granted through a database role that has the `WITH ADMIN` status.

26.6.2.3 Hierarchical System Privileges by Database Account Report

The Hierarchical System Privileges by Database Account Report shows a hierarchical breakdown of role-based system privileges and direct system privileges.

These privileges are granted to the database account specified on the Report Parameters page.

26.6.2.4 ANY System Privileges for Database Accounts Report

The ANY System Privileges for Database Accounts Report shows `ANY` system privileges granted to the specified database account or role.

`ANY` system privileges are very powerful and should be judiciously assigned to accounts and roles.

26.6.2.5 System Privileges By Privilege Report

The System Privileges By Privilege Report lists database accounts and roles that have the system privilege selected on the Report Parameters page.

Another way to control privileges is to create privilege analysis policies to analyze privilege use.

26.6.3 Sensitive Objects Reports

The sensitive objects reports track activities such as grants on the `EXECUTE` privilege on `SYS` schema objects and access to sensitive objects.

- Execute Privileges to Strong SYS Packages Report
  The Execute Privileges to Strong SYS Packages Report shows database accounts and roles with the `EXECUTE` privilege on powerful system packages.
- Access to Sensitive Objects Report
  The Access to Sensitive Objects Report shows database accounts and roles that have object privileges on system tables or views that have sensitive information.
- Public Execute Privilege To SYS PL/SQL Procedures Report
  The Public Execute Privilege to SYS PL/SQL Procedures Report shows database accounts and roles that have `EXECUTE` privileges on that `SYS` owns.
- Accounts with SYSDBA/SYSOPER Privilege Report
  The Accounts with SYSDBA/SYSOPER Privilege Report displays database accounts that have `SYS`-privileged connection privileges.

26.6.3.1 Execute Privileges to Strong SYS Packages Report

The Execute Privileges to Strong SYS Packages Report shows database accounts and roles with the `EXECUTE` privilege on powerful system packages.

For example, these types of packages can be used to access operating system resources.

The following system PL/SQL packages are included:

Packages D-D	Packages D-U
DBMS_ALERT	DBMS_RANDOM
DBMS_BACKUP_RESTORE	DBMS_REPAIR
DBMS_CAPTURE_ADM	DBMS_REPCAT
DBMS_CRYPTO	DBMS_REPCAT_ADMIN
DBMS_DDL	DBMS_RESOURCE_MANAGER

Packages D-D	Packages D-U
DBMS_DISTRIBUTED_TRUST_ADMIN	DBMS_RESOURCE_MANAGER_PRIVS
DBMS_FGA	DBMS_RLS
DBMS_JOB	DBMS_SESSION
DBMS_LDAP	DEBUG_EXTPROC
DBMS_LOB	UTL_FILE
DBMS_LOGMNR	UTL_HTTP
DBMS_LOGMNR_D	UTL_SMTP
DBMS_ORACLE_TRACE_AGENT	UTL_TCP
DBMS_PIPE	-

26.6.3.2 Access to Sensitive Objects Report

The Access to Sensitive Objects Report shows database accounts and roles that have object privileges on system tables or views that have sensitive information.

This report includes the following system tables and views:

Tables/Views A-O	Tables/Views P-S
ALL_SOURCE	PROFILE$
ALL_USERS	PROXY_ROLE_DATA$
APPROLE$	PROXY_ROLE_INFO$
AUD$	ROLE_ROLE_PRIVS
AUDIT_TRAIL$	SOURCE$
DBA_ROLE_PRIVS	STATS$SQLTEXT
DBA_ROLES	STATS$SQL_SUMMARY
DBA_TAB_PRIVS	SYSTEM_PRIVILEGE_MAP
DBMS_BACKUP_RESTORE	TABLE_PRIVILEGE_MAP
DEFROLE$	TRIGGER$
FGA_LOG$	USER$
LINK$	USER_HISTORY$
OBJ$	USER_TAB_PRIVS
OBJAUTH$	SYSTEM_PRIVILEGE_MAP
OBJPRIV$	-

26.6.3.3 Public Execute Privilege To SYS PL/SQL Procedures Report

The Public Execute Privilege to SYS PL/SQL Procedures Report shows database accounts and roles that have `EXECUTE` privileges on that `SYS` owns.

This report can be used to determine which privileges can be revoked from `PUBLIC`, or from other accounts and roles. This reduces vulnerabilities as part of an overall security policy implementation using the principle of least privilege.

26.6.3.4 Accounts with SYSDBA/SYSOPER Privilege Report

The Accounts with SYSDBA/SYSOPER Privilege Report displays database accounts that have `SYS`-privileged connection privileges.

This report also shows whether the accounts use an external password. However, note that this report does not include operating system users who can become `SYSDBA`.

26.6.4 Privilege Management - Summary Reports

The privilege management summary reports track privilege distribution by grantees, owners, and privileges.

- Privileges Distribution By Grantee Report
  The Privileges Distribution By Grantee Report displays the count of privileges granted to a database account or role.
- Privileges Distribution By Grantee, Owner Report
  The Privileges Distribution By Grantee, Owner Report displays a count of privileges based on the grantee and the owner of the object.
- Privileges Distribution By Grantee, Owner, Privilege Report
  The Privileges Distribution By Grantee, Owner, Privilege Report displays a count of privileges based on the privilege, the grantee, and the object owner.

> **See Also:**
>
> DBA_DV_PUB_PRIVS View to find the values on which the counts listed in these reports are based

26.6.4.1 Privileges Distribution By Grantee Report

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

This report provides insight into accounts and roles that may have powerful privileges.

26.6.4.2 Privileges Distribution By Grantee, Owner Report

The Privileges Distribution By Grantee, Owner Report displays a count of privileges based on the grantee and the owner of the object.

This report provides insight into accounts or roles that may have powerful privileges. You can use this report if you suspect potential intruders or insider threats are looking for accounts that have powerful privileges as accounts to attack or compromise. If intruders can compromise the account (for example, by guessing the password), they can get more privileges than they already have.

26.6.4.3 Privileges Distribution By Grantee, Owner, Privilege Report

The Privileges Distribution By Grantee, Owner, Privilege Report displays a count of privileges based on the privilege, the grantee, and the object owner.

This report provides insight into the accounts or roles that may have powerful privileges.

26.6.5 Powerful Database Accounts and Roles Reports

The powerful database accounts and roles reports track information about users who have been granted power privileges, such as the `WITH ADMIN` privilege.

- WITH ADMIN Privilege Grants Report
  The WITH ADMIN Privileges Grants Report shows all database accounts and roles that have been granted privileges with the `WITH ADMIN` clause.
- Accounts With DBA Roles Report
  The Accounts With DBA Roles Report shows all database accounts that have the `DBA` role granted to them.
- Security Policy Exemption Report
  The Security Policy Exemption Report shows database (but not Oracle Database Vault) accounts and roles that have the `EXEMPT ACCESS POLICY` system privilege.
- BECOME USER Report
  The BECOME USER Report shows database accounts roles that have the `BECOME USER` system privilege.
- ALTER SYSTEM or ALTER SESSION Report
  The ALTER SYSTEM or ALTER SESSION Report shows database accounts and roles that have the `ALTER SYSTEM` or `ALTER SESSION` privilege.
- Password History Access Report
  The Password History Access Report shows database accounts that have access to the `USER_HISTORY$` table.
- WITH GRANT Privileges Report
  The WITH GRANT Privileges Report shows database accounts that are granted privileges with the `WITH GRANT` clause.
- Roles/Accounts That Have a Given Role Report
  This report displays the database accounts and roles to which a role has been granted.
- Database Accounts With Catalog Roles Report
  The Database Accounts With Catalog Roles Report displays all database accounts and roles that have the catalog-related roles granted to them.
- AUDIT Privileges Report
  The AUDIT Privileges Report displays all database accounts and roles that have the `AUDIT ANY` or `AUDIT SYSTEM` privilege.
- OS Security Vulnerability Privileges Report
  The OS Security Vulnerability Privileges Report lists database accounts and roles that have privileges to export sensitive information to the operating system.

26.6.5.1 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.

26.6.5.2 Accounts With DBA Roles Report

The Accounts With DBA Roles Report shows all database accounts that have the `DBA` role granted to them.

The `DBA` role is a privileged role that can be misused. It is often granted to a database account to save time and to avoid having to determine the least number of privileges an account really needs. This report can help you to start applying a policy using the principle of least privilege to an existing database.

> **See Also:**
>
> Oracle Database Vault Security Guidelines for guidelines on deciding who should have privileged roles

26.6.5.3 Security Policy Exemption Report

The Security Policy Exemption Report shows database (but not Oracle Database Vault) accounts and roles that have the `EXEMPT ACCESS POLICY` system privilege.

Accounts that have this privilege can bypass all Virtual Private Database (VPD) policy filters and any Oracle Label Security policies that use Oracle Virtual Private Database indirectly. This is a powerful system privilege that should be granted only if absolutely necessary, as it presents a target to gain access to sensitive information in tables that are protected by Oracle Virtual Private Database or Oracle Label Security. You can use the auditing policies described in Auditing Oracle Database Vault, to audit the use of this privilege.

26.6.5.4 BECOME USER Report

The BECOME USER Report shows database accounts roles that have the `BECOME USER` system privilege.

The `BECOME USER` privilege is a very powerful system privilege: it enables the `IMP_FULL_DATABASE` and `EXP_FULL_DATABASE` roles for use with Oracle Data Pump. Accounts that possess this privilege can be misused to get sensitive information or to compromise an application.

26.6.5.5 ALTER SYSTEM or ALTER SESSION Report

The ALTER SYSTEM or ALTER SESSION Report shows database accounts and roles that have the `ALTER SYSTEM` or `ALTER SESSION` privilege.

Oracle recommends that you restrict these privileges only to those accounts and roles that truly need them (for example, the SYS account and the DV_ADMIN role). The ALTER SYSTEM statement can be used to change the security-related database initialization parameters that are set to recommended values as part of the Oracle Database Vault security strengthening service. Both the ALTER SYSTEM and ALTER SESSION statements can be used to dump database trace files, potentially containing sensitive configuration information, to the operating system.

> **See Also:**
>
> ALTER SYSTEM and ALTER SESSION Privilege Security Considerations for guidelines on using the ALTER SYSTEM and ALTER SESSION privileges

26.6.5.6 Password History Access Report

The Password History Access Report shows database accounts that have access to the USER_HISTORY$ table.

This table stores hashed passwords that were previously used by each account.

Access to this table can make guessing the existing password for an account easier for someone hacking the database.

26.6.5.7 WITH GRANT Privileges Report

The WITH GRANT Privileges Report shows database accounts that are granted privileges with the WITH GRANT clause.

Remember that WITH GRANT is used for object-level privileges: An account that has been granted privileges using the WITH GRANT option can be misused to grant object privileges to another account.

26.6.5.8 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.

26.6.5.9 Database Accounts With Catalog Roles Report

The Database Accounts With Catalog Roles Report displays all database accounts and roles that have the catalog-related roles granted to them.

These roles are as follows:

- DELETE_CATALOG_ROLE
- EXECUTE_CATALOG_ROLE
- RECOVERY_CATALOG_OWNER
- SELECT_CATALOG_ROLE

These catalog-based roles have a very large number of powerful privileges. They should be granted with caution, much like the DBA role, which uses them.

26.6.5.10 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.

26.6.5.11 OS Security Vulnerability Privileges Report

The OS Security Vulnerability Privileges Report lists database accounts and roles that have privileges to export sensitive information to the operating system.

This report can reveal important vulnerabilities related to the operating system.

26.6.6 Initialization Parameters and Profiles Reports

The initialization parameters and profiles reports track database parameters, resource profiles, and system limits.

- Security Related Database Parameters Report
  The Security Related Database Parameters Report lists database parameters that can cause security vulnerabilities if they not set correctly.
- Resource Profiles Report
  The Resource Profiles Report lists resource profiles that may be allowing unlimited resource consumption.
- System Resource Limits Report
  The System Resource Limits Report provides insight into the current system resource usage by the database.

26.6.6.1 Security Related Database Parameters Report

The Security Related Database Parameters Report lists database parameters that can cause security vulnerabilities if they not set correctly.

This report can be used to compare the recommended settings with the current state of the database parameter values.

26.6.6.2 Resource Profiles Report

The Resource Profiles Report lists resource profiles that may be allowing unlimited resource consumption.

Examples of resource profiles are `CPU_PER_SESSION` and `IDLE_TIME`. You should review the profiles that might need a cap on the potential resource usage.

26.6.6.3 System Resource Limits Report

The System Resource Limits Report provides insight into the current system resource usage by the database.

This report helps determine whether any of these resources are approaching their limits under the existing application load. Resources that show large increases over a short period may point to a denial-of-service (DoS) attack. You might want to reduce the upper limit for the resource to prevent the condition in the future.

26.6.7 Database Account Password Reports

The database account password reports track default passwords and account statuses of database accounts.

- Database Account Default Password Report
  The Database Account Default Password Report lists the database accounts that have default passwords.
- Database Account Status Report
  The Database Account Status Report lists existing database accounts.

26.6.7.1 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.

26.6.7.2 Database Account Status Report

The Database Account Status Report lists existing database accounts.

This report shows the account status for each account, which helps you identify accounts that must be locked. Lock and expiry dates provide information that helps determine whether the account was locked as a result of password aging. If a special password and resource secure profile is used, then you can identify accounts that are not using them. Accounts not using organizationally defined default tablespaces also can be identified, and the temporary tablespace for accounts can be determined. This report also identifies accounts that use external passwords.

26.6.8 Security Audit Report: Core Database Audit Report

The Core Database Audit Report lists database audit trail records..

This report applies to a non-unified auditing environment.

The Core Database Audit Report returns audit records for the audit policy defined in Auditing Oracle Database Vault, and any auditing records that are generated for audit statements you have defined.

This report only displays audit records that are captured if the database initialization parameter `AUDIT_TRAIL` (deprecated) has been set to `DB` (with unified auditing disabled).

26.6.9 Other Security Vulnerability Reports

Other security vulnerability reports track vulnerabilities that arise with activities such as Java policy grants ir operating system directory objects.

- Java Policy Grants Report
  The Java Policy Grants Report shows the Java policy permissions stored in the database.
- OS Directory Objects Report
  The OS Directory Objects Report shows directory objects in the database, their privileges, and whether they are available to `PUBLIC`.
- Objects Dependent on Dynamic SQL Report
  The Objects Dependent on Dynamic SQL Report lists objects that use dynamic SQL.
- Unwrapped PL/SQL Package Bodies Report
  The Unwrapped PL/SQL Package Bodies Report lists PL/SQL package procedures that are not wrapped.
- Username/Password Tables Report
  The Username/Password Tables Report identifies application tables in the database that store user names and password strings.
- Tablespace Quotas Report
  The Tablespace Quotas Report lists database accounts that have quotas on one or more tablespaces.
- Non-Owner Object Trigger Report
  The Non-Owner Object Trigger Report lists non-owner triggers.

26.6.9.1 Java Policy Grants Report

The Java Policy Grants Report shows the Java policy permissions stored in the database.

This report helps reveal violations to the principle of least privilege. Look for `GRANT`, `READ`, or `WRITE` privileges to `PUBLIC` or other accounts and roles that do not necessarily need the privilege. It is advisable to disable Java loading privileges from `PUBLIC`, if Java is not required in the database.

> **Note:**
>
> Oracle JVM, the Java virtual machine option provided with Oracle Database Vault, must be installed before you can run the Java Policy Grants Report.

26.6.9.2 OS Directory Objects Report

The OS Directory Objects Report shows directory objects in the database, their privileges, and whether they are available to `PUBLIC`.

Directory objects should exist only for secured operating system (OS) directories, and access to them within the database should be protected. You should never use the root operating system directory on any storage device (for example, `/`), because it allows remote database sessions to look at all files on the device.

26.6.9.3 Objects Dependent on Dynamic SQL Report

The Objects Dependent on Dynamic SQL Report lists objects that use dynamic SQL.

Potential intruders have a greater chance of using this channel if parameter checking or bind variables are not used. The report helps by narrowing the scope of where to look for problems by pointing out who is using dynamic SQL. Such objects can be a target for a SQL injection attack and must be secured to avoid this type of attack. After determining the objects that use dynamic SQL, do the following:

- Check the privileges that client applications (for example, a Web application) have over the object.
- Check the access granted for the object to `PUBLIC` or a wider account base.
- Validate parameters.
- Use bind variables where possible.

26.6.9.4 Unwrapped PL/SQL Package Bodies Report

The Unwrapped PL/SQL Package Bodies Report lists PL/SQL package procedures that are not wrapped.

Oracle provides a wrap utility that obfuscates code to the point where it cannot be read in the data dictionary or from the data dictionary views. This helps reduce the ability of an intruder to circumvent data protection by eliminating the ability to read source code that manipulates data.

26.6.9.5 Username/Password Tables Report

The Username/Password Tables Report identifies application tables in the database that store user names and password strings.

You should examine these tables to determine if the information is encrypted. (Search for column names such as `%USER%NAME%` or `%PASSWORD%`.) If it is not, modify the code and applications using these tables to protect them from being visible to database sessions.

26.6.9.6 Tablespace Quotas Report

The Tablespace Quotas Report lists database accounts that have quotas on one or more tablespaces.

These tablespaces can become potential targets for denial-of-service (DoS) attacks.

26.6.9.7 Non-Owner Object Trigger Report

The Non-Owner Object Trigger Report lists non-owner triggers.

These are triggers that are owned by a database account that is different from the account that owns the database object on which the trigger acts.

If the trigger is not part of a trusted database application, then it can *steal* sensitive data, possibly from tables protected through Oracle Label Security or Virtual Private Database (VPD), and place it into an unprotected table for subsequent viewing or export.

A

Auditing Oracle Database Vault

You can audit activities in Oracle Database Vault, such as changes to policy configurations.

- About Auditing in Oracle Database Vault
  All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.
- Oracle Database Vault Specific Audit Events
  Oracle Database Vault traditional (non-unified) audit events track activities such as whether an action attempted on a realm was successful.
- Archiving and Purging the Oracle Database Vault Audit Trail
  If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.
- Oracle Database Vault Predefined Unified Audit Policies
  Oracle Database provides two predefined unified audit policies for Oracle Database Vault: `ORA_DV_SCHEMA_CHANGES` and `ORA_DV_DEFAULT_PROTECTION`.

A.1 About Auditing in Oracle Database Vault

All activities in Oracle Database Vault can be audited, including Database Vault administrator activities.

- Auditing Oracle Database Vault Using Unified Auditing
  Oracle recommends that you migrate all your Oracle Database Vault audit policies to unified auditing.
- Auditing Oracle Database Vault Using Traditional Auditing
  Traditional auditing is desupported, but traditional audit settings in an upgraded database can be used with some limitations.

A.1.1 Auditing Oracle Database Vault Using Unified Auditing

Oracle recommends that you migrate all your Oracle Database Vault audit policies to unified auditing.

To create **new** audit policies in Oracle Database Vault, you must use unified auditing. Traditional auditing is no longer supported as of Oracle Database 23c for new audit settings, though the current existing traditional audit settings are still honored.

The unified audit trail will capture Oracle Database Vault unified audit policy records, as well as mandatory Oracle Database Vault audits.

In addition to this functionality, unified auditing provides the following two predefined policies that are designed for commonly used Database Vault auditing needs:

- `ORA_DV_SCHEMA_CHANGES` (previously called `ORA_DV_AUDPOL`) audits Oracle Database Vault `DVSYS` and `LBACSYS` schema objects.

- `ORA_DV_DEFAULT_PROTECTION` (previously called `ORA_DV_AUDPOL2`) audits the Oracle Database Vault default realms and command rules.

In a new Oracle Database installation, the `ORA_DV_SCHEMA_CHANGES` and `ORA_DV_DEFAULT_PROTECTION` policies are enabled by default. In an upgraded Oracle database, these policies are not enabled by default.

When you use unified auditing, the auditing features in the Database Vault APIs (the `audit_options` parameter) are no longer effective. You should archive and then purge these audit records. From then on, you can manage Database Vault audit policies through the unified audit policy SQL statements.

All configuration changes made to Database Vault are mandatorily audited and these audit records are written to the unified audit trail, including actions of unprivileged users who attempt to modify Database Vault policies.

To learn how to create unified audit policies, including finding examples of Oracle Database Vault unified audit policies, see *Oracle Database Security Guide*.

Related Topics

- Archiving and Purging the Oracle Database Vault Audit Trail
  If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.

A.1.2 Auditing Oracle Database Vault Using Traditional Auditing

Traditional auditing is desupported, but traditional audit settings in an upgraded database can be used with some limitations.

The traditional audit settings are controlled through the `audit_options` parameter when you create or modify realms, rule sets, and factors. The audit indicates if the user's action succeeded (that is, the policy enabled the user to accomplish a task) or if the user's action failed (the policy was violated). Database Vault APIs use this parameter to collect audit records and write these audit records to the Oracle Database Vault data dictionary views and reports. Though traditional auditing is desupported starting in Oracle Database release 23c, any current existing Oracle Database Vault traditional audit settings that you have will still be honored. However, you cannot create new traditional audit settings. You can delete existing traditional audit settings. For example, suppose you created a realm in an earlier release and this realm uses traditional auditing. If you want to update the audit settings for the realm, then the only option available for the `audit_options` parameter is the `OFF` option. To track auditing for the realm, create a unified audit policy. If you do not update the `audit_options` parameter, that is, if you do not update the realm, or if you use `NULL` or the same `audit_options` setting as the current existing setting when updating the realm, then the traditional audit settings defined by `audit_options` will remain as is and continue to work.

When you install a new database and configure it to use Oracle Database Vault, then by default it uses unified auditing only. If you have upgraded from a previous release, then Database Vault uses the auditing that was available from that release.

See *Oracle Database Security Guide* for more information about how the desupport of traditional auditing works.

Related Topics

- Archiving and Purging the Oracle Database Vault Audit Trail
  If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.

A.2 Oracle Database Vault Specific Audit Events

Oracle Database Vault traditional (non-unified) audit events track activities such as whether an action attempted on a realm was successful.

- Oracle Database Vault Policy Audit Events
  Oracle Database Vault uses audit events to track configuration activities, using traditional, non-unified auditing.
- Oracle Database Vault Audit Trail Record Format
  The `DVSYS.AUDIT_TRAIL$` table stores traditional audit trail records.

A.2.1 Oracle Database Vault Policy Audit Events

Oracle Database Vault uses audit events to track configuration activities, using traditional, non-unified auditing.

These activities are as follows:

- **Realm Audit.** You can audit both successful and failed actions, based on the auditing option that you set when you created the realm. The exception to this is actions performed by the schema owner.
- **Rule Set Audit.** Audits the rule set processing results. You can audit both successful and failed processing. Realm authorizations can be managed using rule sets. You can audit the rule set processing results. Factor assignments and secure application roles audits can be managed using a rule set.
- **Factor Audit.** You can audit both successful and failed factor processing. For failed factor processing, you can audit on all or any of the following events: Retrieval Error, Retrieval Null, Validation Error, Validation False, Trust Level Null, or Trust Level Less Than Zero.
- **Oracle Label Security Session Initialization Failed.** Audits instances where the Oracle Label Security session fails to initialize.
- **Oracle Label Security Attempt to Upgrade Session Label Failed.** Audits instances where the Oracle Label Security component prevents a session from setting a label that exceeds the maximum session label.

Related Topics

- 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.
- About Realm Authorization
  Realm authorizations establish the set of database accounts and roles that manage or access objects protected in realms.
- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.

A.2.2 Oracle Database Vault Audit Trail Record Format

The DVSYS.AUDIT_TRAIL$ table stores traditional audit trail records.

Starting in Oracle Database 23c, traditional auditing is desupported. However, traditional audit settings that are currently in place will still be honored. The audit_options settings from realms, rule sets, or factors from a previous release will continue to write to the DVSYS.AUDIT_TRAIL$ table.

Users who have been granted the DV_OWNER, DV_ADMIN, DV_SECANALYST or DV_MONITOR role can directly query the DVSYS.AUDIT_TRAIL$ table.

Table A-1 describes the format of the DVSYS.AUDIT_TRAIL$ table.

Table A-1 Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
ID#	NUMBER	NOT NULL	Numeric identifier for the audit record
OS_USERNAME	VARCHAR2(255)	NULL	Operating system login user name of the user whose actions were audited
USERNAME	VARCHAR2(30)	NULL	Name of the database user whose actions were audited
USERHOST	VARCHAR2(128)	NULL	Client computer name
TERMINAL	VARCHAR2(255)	NULL	Identifier for the user's terminal
TIMESTAMP	DATE	NULL	Date and time of creation of the audit trail entry (in the local database session time zone)
OWNER	VARCHAR2(30)	NULL	Creator of the object affected by the action, always DVSYS (because DVSYS is where objects are created)
OBJ_NAME	VARCHAR2(128)	NULL	Name of the object affected by the action. Expected values are: • ROLE$ • REALM$ • CODE$ • FACTOR$
ACTION	NUMBER	NOT NULL	Numeric action type code. The corresponding name of the action type is in the ACTION_NAME column. See Table 24-3 for a list of the expected ACTION and ACTION_NAME values.
ACTION_NAME	VARCHAR2(128)	NULL	Name of the action type corresponding to the numeric code in the ACTION column
ACTION_OBJECT_ID	NUMBER	NULL	The unique identifier of the record in the table specified under OBJ_NAME. For realms, this field contains a list of comma-separated values of all realm IDs that have the Audit on Failure audit option.

Table A-1 (Cont.) Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
ACTION_OBJECT_NAME	VARCHAR2(128)	NULL	The unique name or natural key of the record in the table specified under OBJ_NAME. For realms, this field contains a list of comma-separated values of all realm names that have the Audit on Failure audit option.
ACTION_COMMAND	VARCHAR2(4000)	NULL	The SQL text of the command procedure that was run that resulted in the audit event being triggered
AUDIT_OPTION	VARCHAR2(4000)	NULL	The labels for all audit options specified in the record that resulted in the audit event being triggered. For example, a factor set operation that is supposed to audit on get failure and get NULL would indicate these two options.
RULE_SET_ID	NUMBER	NULL	The unique identifier of the rule set that was executing and caused the audit event to trigger
RULE_SET_NAME	VARCHAR2(30)	NULL	The unique name of the rule set that was executing and caused the audit event to trigger
RULE_ID	NUMBER	NULL	Not used
RULE_NAME	VARCHAR2(30)	NULL	Not used
FACTOR_CONTEXT	VARCHAR2(4000)	NULL	An XML document that contains all of the factor identifiers for the current session at the point when the audit event was triggered
COMMENT_TEXT	VARCHAR2(4000)	NULL	Text comment on the audit trail entry, providing more information about the statement audited
SESSIONID	NUMBER	NOT NULL	Numeric identifier for each Oracle session
ENTRYID	NUMBER	NOT NULL	Same as the value in the ID# column
STATEMENTID	NUMBER	NOT NULL	Numeric identifier for the statement invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
RETURNCODE	NUMBER	NOT NULL	Oracle error code generated by the action. The error code for a statement or procedure invoked that caused the audit event to be generated. This is empty for most Oracle Database Vault events.
EXTENDED_TIMESTAMP	TIMESTAMP(6) WITH TIME ZONE	NULL	Time stamp of creation of the audit trail entry (time stamp of user login for entries) in UTC (Coordinated Universal Time) time zone
PROXY_SESSIONID	NUMBER	NULL	Proxy session serial number, if an enterprise user has logged in through the proxy mechanism
GLOBAL_UID	VARCHAR2(32)	NULL	Global user identifier for the user, if the user has logged in as an enterprise user
INSTANCE_NUMBER	NUMBER	NULL	Instance number as specified by the INSTANCE_NUMBER initialization parameter
OS_PROCESS	VARCHAR2(16)	NULL	Operating system process identifier of the Oracle process
CREATED_BY	VARCHAR2(30)	NULL	Database login user name of the user whose actions were audited

Table A-1 (Cont.) Oracle Database Vault Audit Trail Format

Column	Datatype	Null	Description
CREATE_DATE	DATE	NULL	Date on which the action occurred, based on the SYSDATE date
UPDATED_BY	VARCHAR2(30)	NULL	Same as CREATED_BY column value
UPDATE_DATE	DATE	NULL	Same as UPDATED_BY column value

A.3 Archiving and Purging the Oracle Database Vault Audit Trail

If you have not migrated to unified auditing, you should periodically archive and purge the Oracle Database Vault audit trail.

- About Archiving and Purging the Oracle Database Vault Audit Trail
  In a traditional, non-unified auditing environment, you can archive the Oracle Database Vault audit trail by exporting the DVSYS.AUDIT_TRAIL$ table to a dump file.
- Archiving the Oracle Database Vault Audit Trail
  You can use SQL*Plus and Oracle Data Pump to archive the Oracle Database Vault audit trail from the root or a PDB.
- Purging the Oracle Database Vault Audit Trail
  You can purge the (traditional, non-unified auditing) Oracle Database Vault audit trail from the root or a PDB.

A.3.1 About Archiving and Purging the Oracle Database Vault Audit Trail

In a traditional, non-unified auditing environment, you can archive the Oracle Database Vault audit trail by exporting the DVSYS.AUDIT_TRAIL$ table to a dump file.

You should periodically archive and then purge the audit trail to prevent it from growing too large.

If you choose to migrate to unified auditing, then use this procedure to archive and purge the Database Vault audit trail records after you complete the migration. When unified auditing begins to collect records, then the new records will be available for viewing from the UNIFIED_AUDIT_TRAIL, AUDSYS.DV$CONFIGURATION_AUDIT, and AUDSYS.DV$ENFORCEMENT_AUDIT data dictionary views.

A.3.2 Archiving the Oracle Database Vault Audit Trail

You can use SQL*Plus and Oracle Data Pump to archive the Oracle Database Vault audit trail from the root or a PDB.

Use this procedure to archive the traditional, non-unified audit trail in Oracle Database Vault.

1. As user `SYS` with the `SYSDBA` administrative privilege, log in to the root or to the PDB.
2. Ensure that the user who will perform archiving has the appropriate privileges.

   For example:

   ```
   GRANT CREATE ANY DIRECTORY, EXP_FULL_DATABASE, UNLIMITED TABLESPACE TO psmith;
   ```

3. Connect to the root or the PDB as a user who has been granted the `DV_OWNER` or `DV_AUDIT_CLEANUP` role.
4. Ensure that the user who will perform archiving has the appropriate privileges.

   ```
   GRANT CREATE ANY DIRECTORY, EXP_FULL_DATABASE, UNLIMITED TABLESPACE TO user_name;
   ```

5. Connect to the root or the PDB as a user who has been granted the `DV_OWNER` or `DV_AUDIT_CLEANUP` role.
6. Archive the Oracle Database Vault audit trail into a new table in an appropriate schema.

   For example:

   ```
   CREATE TABLE psmith.dv_audit_trail nologging \
   AS SELECT * FROM DVSYS.AUDIT_TRAIL$;
   ```

7. If the schema is already protected by a realm, then ensure that you or the user performing the export operation has been granted the appropriate authorization to use Oracle Data Pump in a Database Vault environment.

   For example, to authorize user `psmith` to perform Data Pump operations on their own schema:

   ```
   EXEC DBMS_MACADM.AUTHORIZE_DATAPUMP_USER('PSMITH', 'PSMITH');
   ```

8. Connect to the root or the PDB as the Data Pump user.
9. Create a directory for the Database Vault audit trail.

   ```
   CREATE DIRECTORY dv_audit_dir AS 'dv_audit_trail_directory';
   ```

10. Exit SQL*Plus.

    ```
    EXIT
    ```

11. Using Data Pump, export the Database Vault audit trail into the directory object that you just created.

    ```
    expdp psmith directory=dv_audit_dir tables=psmith.dv_audit_trail \
    dumpfile=dv_audit.dmp log=dv_audit_exp.log
    ```

12. Connect to the root or the PDB as a user who has been granted the `DV_OWNER` role.
13. If you have not done so, then create a realm around the schema that now contains the Database Vault audit trail.
    a. Create the realm. For example:

       ```
       BEGIN
        DBMS_MACADM.CREATE_REALM(
         realm_name    => 'DV Audit Trail Realm',
         description   => 'Realm to protect the DV audit trail',
         enabled       => DBMS_MACUTL.G_YES,
         audit_options => DBMS_MACUTL.G_REALM_AUDIT_ON,
         realm_type    => 1);
       END;
       /
       ```

    b. Add the schema that contains to audit trail to this realm. For example:

```
BEGIN
 DBMS_MACADM.ADD_OBJECT_TO_REALM(
  realm_name   => 'DV Audit Trail Realm',
  object_owner => 'psmith',
  object_name  => '%',
  object_type  => '%');
END;
/
```

c. Authorize a trusted user for this realm.

```
BEGIN
 DBMS_MACADM.ADD_AUTH_TO_REALM(
  realm_name  => 'DV Audit Trail Realm',
  grantee     => 'PSMITH',
  auth_options => DBMS_MACUTL.G_REALM_AUTH_OWNER);
END;
/
```

Related Topics

- Using Oracle Data Pump with Oracle Database Vault
  Database administrators can authorize Oracle Data Pump users to work in a Database Vault environment.
- *Oracle Database SQL Language Reference*
- *Oracle Database Utilities*

A.3.3 Purging the Oracle Database Vault Audit Trail

You can purge the (traditional, non-unified auditing) Oracle Database Vault audit trail from the root or a PDB.

1. As user who has been granted the `DV_OWNER` role or the `DV_AUDIT_CLEANUP` role, log in to the root or to the PDB.
2. Purge the Database Vault audit trail.

```
DELETE FROM DVSYS.AUDIT_TRAIL$;
```

Related Topics

- DV_AUDIT_CLEANUP Audit Trail Cleanup Role
  The `DV_AUDIT_CLEANUP` role is used for purge operations.

A.4 Oracle Database Vault Predefined Unified Audit Policies

Oracle Database provides two predefined unified audit policies for Oracle Database Vault: `ORA_DV_SCHEMA_CHANGES` and `ORA_DV_DEFAULT_PROTECTION`.

- `ORA_DV_SCHEMA_CHANGES` audits Oracle Database Vault `DVSYS` and `LBACSYS` schema objects. See *Oracle Database Security Guide* for detailed information about this predefined policy.
- `ORA_DV_DEFAULT_PROTECTION` audits the Oracle Database Vault default realms and command rules. See *Oracle Database Security Guide* for detailed information about this predefined policy.

You can find the definitions of these policies by querying the `AUDIT_UNIFIED_POLICIES` data dictionary view. For the `policy_name` parameter, specify `ORA_DV_SCHEMA_CHANGES` or `ORA_DV_DEFAULT_PROTECTION`.

B

Disabling and Enabling Oracle Database Vault

Periodically you must disable and then re-enable Oracle Database Vault, for activities such as installing Oracle Database optional products or features.

- When You Must Disable Oracle Database Vault
  You may need to disable Oracle Database Vault to perform upgrade tasks or correct erroneous configurations.
- Step 1: Disable Oracle Database Vault
  Be aware that after you disable Oracle Database Vault, Oracle Label Security, which is required to run Database Vault, is still enabled.
- Step 2: Perform the Required Tasks
  At this stage, Oracle Database Vault is disabled and you can perform the required tasks.
- Step 3: Enable Oracle Database Vault
  You can enable Oracle Database Vault and Oracle Label Security from SQL*Plus from either the root or a PDB.

B.1 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.

The following situations require you to disable Oracle Database Vault:

- You must install any of the Oracle Database optional products or features, such as Oracle Spatial, by using Database Configuration Assistant (DBCA).
- If you did not configure backup `DV_OWNER` and `DV_ACCTMGR` accounts when you configured and enabled Oracle Database Vault, and these accounts are inadvertently locked or their passwords forgotten. Note that if your site only has one `DV_OWNER` user and this user has lost their password, you will be unable to disable Oracle Database Vault. However, if your site's only `DV_ACCTMGR` user has lost the password, you can disable Database Vault. As a best practice, you should grant the `DV_OWNER` and `DV_ACCTMGR` roles to new or existing user accounts, and use the Database Vault Owner and Account Manager accounts that you created when you configured and enabled Database Vault as back-up accounts.
- If you want to configure Oracle Internet Directory (OID) using Oracle Database Configuration Assistant (DBCA).
- If Oracle Database Vault is enabled and you are upgrading an entire CDB, then use one of the following methods:
  - **CDB upgrade method 1:** Temporarily grant the `DV_PATCH_ADMIN` to user `SYS` commonly by logging into the root container as a common user with the `DV_OWNER` role, and then issuing the `GRANT DV_PATCH_ADMIN TO SYS CONTAINER=ALL` statement. Oracle Database Vault controls will be in the same state as it was before the upgrade. When the upgrade is complete, log into the root container as the `DV_OWNER`

user and revoke the DV_PATCH_ADMIN role from SYS by issuing the REVOKE DV_PATCH_ADMIN FROM SYS CONTAINER=ALL statement.

- **CDB upgrade method 2:** Log into each container as a user who has the DV_OWNER role and then run the DBMS_MACADM.DISABLE_DV procedure. You must first disable the PDBs (in any order) and then after that, disable the root container last. If you are upgrading only one PDB, then you can disable Oracle Database Vault in that PDB only. After you have completed the upgrade, you can enable Oracle Database Vault by logging into each container as the DV_OWNER user and then executing the DVSYS.DBMS_MACADM.ENABLE_DV procedure. The order of enabling Oracle Database Vault must be the root container first and PDBs afterward. You can enable the PDBs in any order, but the root container must be enabled first.

> **Note:**
>
> Be aware that if you disable Oracle Database Vault, the privileges that were revoked from existing users and roles during the Oracle Database Vault configuration remain in effect.

Related Topics

- Verifying That Database Vault Is Configured and Enabled
  The DBA_DV_STATUS, CDB_DV_STATUS, and DBA_OLS_STATUS data dictionary views verify if Oracle Database is configured and enabled.
- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the DV_OWNER and DV_ACCTMGR roles.
- Privileges That Are Revoked from Existing Users and Roles
  The Oracle Database Vault configuration revokes privileges from several Oracle Database-supplied users and roles, for better separation of duty.

B.2 Step 1: Disable Oracle Database Vault

Be aware that after you disable Oracle Database Vault, Oracle Label Security, which is required to run Database Vault, is still enabled.

1. As a user who has been granted the DV_OWNER role, log in to the root or to the PDB in which you want to disable Oracle Database Vault.
2. If necessary, verify the enablement status of Oracle Database Vault.
3. Disable Oracle Database Vault.

   ```
   EXEC DBMS_MACADM.DISABLE_DV;
   ```

4. Restart the CDB or close and then reopen the PDB.

   To restart the CDB from the root:

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

   SQL> SHUTDOWN IMMEDIATE
   SQL> STARTUP
   ```

To close and reopen the PDB:

```
CONNECT sec_admin_owen@pdb_name
Enter password: password

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

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

Related Topics

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

B.3 Step 2: Perform the Required Tasks

At this stage, Oracle Database Vault is disabled and you can perform the required tasks.

You can perform the following types of activities:

- **Use the Oracle Database Vault PL/SQL packages and functions.** For example, to correct a login or CONNECT rule set error, use the DBMS_MACADM PL/SQL package or the Oracle Database Vault pages in Enterprise Manager Cloud Control. Note that a CONNECT command rule cannot prevent a user who has the DV_OWNER or DV_ADMIN role from connecting to the database. This enables a Database Vault administrator to correct a misconfigured protection without having to disable Database Vault.
- **Use the SYSTEM or SYS accounts to perform tasks such as creating or changing passwords, or locking and unlocking accounts.** In addition to modifying standard database and administrative user accounts, you can modify passwords and the lock status of any of the Oracle Database Vault-specific accounts, such as users who have been granted the DV_ADMIN or DV_ACCTMGR roles.
- **Perform the installation or other tasks that require security protections to be disabled.**

B.4 Step 3: Enable Oracle Database Vault

You can enable Oracle Database Vault and Oracle Label Security from SQL*Plus from either the root or a PDB.

1. As a user who has been granted the DV_OWNER role, log in to the root or to the PDB in which you want to enable Oracle Database Vault.
2. If necessary, verify the enablement status of Oracle Database Vault.
3. Enable Database Vault.

```
EXEC DBMS_MACADM.ENABLE_DV (strict_mode => 'n');
 -- For regular mode
EXEC DBMS_MACADM.ENABLE_DV (strict_mode => 'y');
 -- For strict mode
```

4. Check if Oracle Label Security is enabled.

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

Oracle Label security must be enabled before you can use Database Vault. If it is not enabled, then this query returns `FALSE`.

5. If Oracle Label Security is not enabled, then enable it.

```
EXEC LBACSYS.CONFIGURE_OLS;
EXEC LBACSYS.OLS_ENFORCEMENT.ENABLE_OLS;
```

6. Restart the CDB or close and then reopen the PDB.

   To restart the CDB from the root:

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

SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

   To close and reopen the PDB:

```
CONNECT sec_admin_owen@pdb_name
Enter password: password

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

7. For Oracle RAC installations, repeat these steps for each node on which the database is installed.

Related Topics

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

C

Postinstallation Oracle Database Vault Procedures

After you configure and enable Oracle Database Vault, you can perform specialized tasks, such as adding languages or uninstalling and reinstalling Oracle Database Vault.

- Adding Languages to Oracle Database Vault
  By default, Oracle Database Vault loads only the English language tables.
- Uninstalling Oracle Database Vault
  You can uninstall Oracle Database Vault from an Oracle Database installation, for PDBs (but not the root) and Oracle RAC installations.
- Reinstalling Oracle Database Vault
  You can reinstall Oracle Database Vault by manually installing it, and then afterward, configure and enable it.

Related Topics

- Converting a Standalone Oracle Database to a PDB and Plugging It into a CDB
  You can convert a standalone Oracle Database database from release 12c through 19c to a PDB, and then plug this PDB into a CDB.

C.1 Adding Languages to Oracle Database Vault

By default, Oracle Database Vault loads only the English language tables.

You can add more languages by running the `DBMS_MACADM.ADD_NLS_DATA` procedure for each new language that you want to add. You can add more than one language to Database Vault, to either a specific PDB or to the root for all PDBs.

1. Log into the root or the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Run the following procedure:

   ```
   EXEC DBMS_MACADM.ADD_NLS_DATA('language');
   ```

   You can specify the *`language`* setting using any case. For example:

   ```
   EXEC DBMS_MACADM.ADD_NLS_DATA('french');

   EXEC DBMS_MACADM.ADD_NLS_DATA('JAPANESE');
   ```

   Replace *`language`* with one of the following supported languages:

   - `ENGLISH`
   - `GERMAN`
   - `SPANISH`
   - `FRENCH`
   - `ITALIAN`

- JAPANESE
- KOREAN
- BRAZILIAN PORTUGUESE
- SIMPLIFIED CHINESE
- TRADITIONAL CHINESE

C.2 Uninstalling Oracle Database Vault

You can uninstall Oracle Database Vault from an Oracle Database installation, for PDBs (but not the root) and Oracle RAC installations.

The uninstallation process does not affect the initialization parameter settings, even those settings that were modified during the installation process, nor does it affect Oracle Label Security.

1. Connect to the PDB as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role.
2. Run the following procedure to disable Oracle Database Vault:

   ```
   EXEC DBMS_MACADM.DISABLE_DV;
   ```

3. Close and reopen the PDB, or for Oracle RAC, restart the database.

   As a user who has the `ALTER PLUGGABLE DATABASE` privilege:

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

   For Oracle RAC installations, shut down and then restart each database instance as follows:

   ```
   srvctl stop database -db db_name
   srvctl start database -db db_name
   ```

4. Run the `dvremov.sql` script to remove Oracle Database Vault.

   For example:

   ```
   $ORACLE_HOME/rdbms/admin/dvremov.sql
   ```

5. If necessary, in SQL*Plus, as user `SYS` with the `SYSDBA` administrative privilege, manually revoke the `EXECUTE` privilege on the `DBMS_RLS` PL/SQL package from any users who have been granted the `DV_OWNER` role.

   When you configure Oracle Database Vault, one of the privileges that `DV_OWNER`users are granted is this privilege. However, when you remove Oracle Database Vault, `DV_OWNER` users still have this privilege. Optionally, you can revoke it.

   ```
   REVOKE EXECUTE ON DBMS_RLS FROM dbv_owner_backup;
   ```

Afterward, you can double-check that Oracle Database Vault is truly deinstalled by logging in to SQL*Plus and entering the following statement:

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

If Oracle Database Vault is deinstalled, the following output appears:

```
PARAMETER                      VALUE
------------------------------ ----------------------
Oracle Database Vault          FALSE
```

C.3 Reinstalling Oracle Database Vault

You can reinstall Oracle Database Vault by manually installing it, and then afterward, configure and enable it.

Related Topics

- Manually Installing Oracle Database Vault
  Under certain conditions, you must manually install Oracle Database Vault.
- Configuring and Enabling Oracle Database Vault
  You can configure and enable Oracle Database Vault based on several scenarios.

D

Oracle Database Vault Security Guidelines

As with all Oracle Database products, you should follow security guidelines to better secure your Oracle Database Vault installation.

- Separation of Duty Guidelines
  Oracle Database Vault is designed to easily implement separation of duty guidelines.
- Managing Oracle Database Administrative Accounts
  Oracle provides guidelines for managing security for administrative accounts such as `SYSTEM` or users who have the `SYSDBA` administrative privilege.
- Accounts and Roles Trusted by Oracle Database Vault
  Oracle Database Vault restricts access to application data from many privileged users and roles in the database.
- Accounts and Roles That Should be Limited to Trusted Individuals
  You should limit powerful accounts and roles only to trusted individuals.
- Guidelines for Using Oracle Database Vault in a Production Environment
  You should follow special guidelines when you run Oracle Database Vault in a production environment.
- Secure Configuration Guidelines
  You should be aware of security considerations for special PL/SQL packages, privileges, and the recycle bin.

D.1 Separation of Duty Guidelines

Oracle Database Vault is designed to easily implement separation of duty guidelines.

- How Oracle Database Vault Handles Separation of Duty
  Separation of duty is restricting each user's privileges *only* to the tasks they are responsible for, and *no more*.
- Separation of Tasks in an Oracle Database Vault Environment
  Oracle Database Vault defines the several main responsibilities.
- Separation of Duty Matrix for Oracle Database Vault
  Before applying separation of duty, you must understand who performs basic administration tasks in your environment and what these administration tasks are.
- Identification and Documentation of the Tasks of Database Users
  You should document the areas of the tasks that your organization needs.

D.1.1 How Oracle Database Vault Handles Separation of Duty

Separation of duty is restricting each user's privileges *only* to the tasks they are 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.

D.1.2 Separation of Tasks in an Oracle Database Vault Environment

Oracle Database Vault defines the several main responsibilities.

These responsibilities are as follows:

- **Account management.** Account management entails creating, modifying, and dropping user accounts. The `DV_ACCTMGR` role provides these privileges. A primary day-to-day `DV_ACCTMGR` user and a backup `DV_ACCTMGR` user are created during the Oracle Database Vault registration process. As a safety measure, you keep and maintain the backup account in case the primary `DV_ACCTMGR` account owner forgets their password or leaves the company.
- **Security administration.** Security administration covers basic security tasks such as creating realms and command rules, setting security policies for database users' access, and authorizing database users for jobs they are allowed to perform. Security administrators also run security audit reports. The `DV_OWNER` and `DV_ADMIN` roles provide these privileges. A primary day-to-day `DV_OWNER` user and a backup `DV_OWNER` user are created during the Oracle Database Vault registration process.

  > **Important:**
  >
  > As a safety measure, you should keep and maintain the backup user account in case the primary `DV_OWNER` account owner forgets their password or leaves the company. It is also important that you do not lose access to all of the user accounts that have been granted the `DV_OWNER` role. There is no way to recover the `DV_OWNER` role if you lose access (such as with a lost password or a staff departure) to any account that has the `DV_OWNER` role. If you lose access to the `DV_OWNER` role, then you cannot modify any Database Vault controls or disable Database Vault. To remedy this problem, you can recover the database to the last known point where the database had possession of the Database Vault owner account.

  Optionally, you can consolidate the account management and security administrative responsibilities.
- **Database management.** Database management refers to managing the database system but not accessing business data. It includes the following operations:
  - Backup operations require a predefined time to perform the backup using predefined tools.

- Tuning and monitoring operations require ongoing performance monitoring and analysis.
- Patching operations require temporary access only during the time the patching takes place

Oracle strongly recommends that you review database management accounts within the context of separation of duty. Different database administrators may have different responsibilities that require different privileges and roles. Similarly, more experienced database administrators may have more roles and privileges. Instead of granting users the default `DBA` role to users, consider tailoring database administrative roles for specific positions and for seniority in your organization. It is important to use only named accounts for day-to-day activities. Accounts such as `SYS` and accounts that use the `SYSDBA` administrative privilege should be managed with Privileged Account Management (PAM) systems and checked out (and audited) when they are used. You should also manage the backup Oracle Database Vault owner and account management accounts with a PAM system. Within the operating system, you should make the `root` and `oracle` accounts available only through a checkout system, because of the powerful privileges that these accounts have.

You should have separate accounts for database account management, database security administration, and additional named accounts for backup operations. Auditors check for separate database accounts for different responsibilities and being able to track the actions of each account. Less important is the number of users assigned to specific tasks. Remember that Oracle Database Vault audit events are protected and that the Database Vault reports show all attempted violations.

Related Topics

- Oracle Database Vault Roles
  Oracle Database Vault provides default roles that are based on specific user tasks and adhere to separation of duty concepts.
- Database Accounts Used by Oracle Database Vault
  Oracle Database Vault provides accounts that provide access to system and object privileges, and Oracle Label Security.
- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the `DV_OWNER` and `DV_ACCTMGR` roles.

D.1.3 Separation of Duty Matrix for Oracle Database Vault

Before applying separation of duty, you must understand who performs basic administration tasks in your environment and what these administration tasks are.

Even if a single database administrator is responsible for managing both new database account provisioning and application patching, it is important to document and plan for each of these tasks. Using separate administration accounts for these types of tasks provides increased accountability and reduces associated risks if and when a single account is compromised by a malicious user. In midsize to large organizations, database administrators typically must perform common administration tasks but they do not need access to business data managed by the application. Creating a matrix for your separation of duty can help you plan your Database Vault deployment. As needed, you can include additional tasks and associated users to this list. This information should become part of the overall enterprise security documentation for your organization.

Table D-1 shows an example of a separation of duty matrix.

Table D-1 Example Separation of Duty Matrix

User, Process or Application	Account Creation	Database Administration					Security Administrator
		SYSDBA	Backup	Tuning	Patching	Monitoring	
JSMITH	Yes	No	No	No	No	No	No
SHARDY	No	No	No	No	No	No	Yes
PKESTNER	No	No	Yes	No	No	No	No
RTYLER	No	No	No	No	Yes	No	No
SANDERSON	No	No	No	Yes	No	Yes	No
SYSTEM	No	No	No	No	Yes, for EBS patching	No	No
RMAN	No	Yes	Yes	No	No	No	No

In some cases, system management tasks may require temporary access to data through specific tools and programs. When this happens, build provisions for this temporary or emergency access into the Oracle Database Vault rules and rule sets.

D.1.4 Identification and Documentation of the Tasks of Database Users

You should document the areas of the tasks that your organization needs.

These areas are as follows:

- The responsibilities of each administrative user
- The kind of access users need. For example, application owners should have data access and developers need access to development instances only.
- Who must manage the system without accessing business data (for example, users who perform backup, patching, tuning, and monitoring operations)
- The duties of each category of tasks (for example, the files that must be backed up, the applications that require patching, what exactly is monitored). Include the alternate user accounts for each of these tasks.
- The databases and applications that must be protected. This includes Oracle applications, partner applications, and custom applications.
- Who must be authorized to access business data, including the following:
  - Application owners through middle tier processes
  - Business users through an application interface
- Emergency "what if" scenarios, such as how to handle a security breach
- Reporting in a production environment, which should include the following:
  - Who runs the reports
  - Which reports must be run
  - The frequency with which each report is run
  - The users who must receive a copy of each report

- In addition to a separation of duty matrix, the creation of the following matrices:
  - An Oracle Database Vault-specific matrix, which can cover the names and tasks of users who have been granted Database Vault roles
  - An application protection matrix, which can cover the applications to be protected and the types of protections you have put in place.

Table D-2 shows an example of protections Oracle created for PeopleSoft Applications. SYSADM, PSFTDBA, SYSTEM, and DBA have all been authorized for the appropriate rule sets.

Table D-2 Example Application Protection Maxtrix

Protection Type	SYSADM	PSFTDBA	SYSTEM	DBA
PeopleSoft Realm	Owner	Owner	No Access	No Access
SELECT Command Rule	Not Restricted	Limit PSFTDB Rule Set	No Access	No Access
CONNECT Command Rule	PeopleSoftAccess Rule Set	Not Restricted	Not Restricted	Not Restricted
DROP TABLESPACE Command Rule	Disabled Rule Set	Disabled Rule Set	Disabled Rule Set	Disabled Rule Set

D.2 Managing Oracle Database Administrative Accounts

Oracle provides guidelines for managing security for administrative accounts such as SYSTEM or users who have the SYSDBA administrative privilege.

- SYSTEM User Account for General Administrative Uses
  Ideally, the SYSTEM account should only be available as a backup that is checked out and audited while being used.
- SYSTEM Schema for Application Tables
  If you have application tables in the SYSTEM schema, then you should add the SYSTEM account to your realm authorizations for these tables.
- Limitation of the SYSDBA Administrative Privilege
  Limit the SYSDBA administrative privilege to users who must connect using this privilege when absolutely necessary and for applications that still require SYSDBA access.
- Root and Operating System Access to Oracle Database Vault
  For better security, you should carefully monitor root and operating system access to Oracle Database Vault.

D.2.1 SYSTEM User Account for General Administrative Uses

Ideally, the SYSTEM account should only be available as a backup that is checked out and audited while being used.

Only named accounts should be used for normal database administration tasks - not shared accounts. Doing so increases accountability for administrative actions in the database.

D.2.2 SYSTEM Schema for Application Tables

If you have application tables in the `SYSTEM` schema, then you should add the `SYSTEM` account to your realm authorizations for these tables.

This enables these applications to continue to work normally.

You can place restrictions on the `SYSTEM` account to increase or fine-tune security for these applications. For example, you can create a Database Vault rule set to restrict the `SYSTEM` user's access to specific IP addresses.

D.2.3 Limitation of the SYSDBA Administrative Privilege

Limit the `SYSDBA` administrative privilege to users who must connect using this privilege when absolutely necessary and for applications that still require `SYSDBA` access.

For example, mandatory patching processes require `SYSDBA` access.

For all other cases, create named database accounts to perform daily database administration. Members of the `OSDBA` user group are also given the `SYSDBA` administrative privilege. The database `SYS` account and accounts with `SYSDBA` privilege along with the operating system `root` and `oracle` accounts should be managed in a Privileged Account Management (PAM) system and checked out only when required.

Related Topics

- Management of SYSDBA Access
  You should avoid using the `SYS` account and the `SYSDBA` privilege for normal database maintenance tasks.

D.2.4 Root and Operating System Access to Oracle Database Vault

For better security, you should carefully monitor root and operating system access to Oracle Database Vault.

Oracle Database Vault prevents highly privileged database users from accessing sensitive data. In addition, if you are using Oracle Database itself, then you can use Transparent Data Encryption to prevent the most highly privileged operating system users from accessing sensitive data. Transparent data encryption enables you to encrypt tablespaces and table columns. This prevents operating system users from browsing through the operating system database files and finding sensitive data. As a best practice, always carefully review and restrict direct access to the operating system.

You should have personalized accounts access the operating system. These personalized accounts should, in the Linux or UNIX environments, login using `sudo` to the `oracle` software owner when needed. With `sudo`, you can control which specific command each personalized user can run. 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.

Related Topics

- *Oracle Database Advanced Security Guide*

D.3 Accounts and Roles Trusted by Oracle Database Vault

Oracle Database Vault restricts access to application data from many privileged users and roles in the database.

However, in some cases, Oracle Database Vaults trusts certain roles and privileges.

Table D-3 lists the trusted roles and privileges that are created when you install Oracle Database Vault.

Table D-3 Trusted Oracle Database Vault Roles and Privileges

Role or Privilege	Status	Description
`DV_ACCTMGR` role	Open	Role created during registration and used for creating new database accounts. As a safety measure, maintain a backup user who has the `DV_ACCTMGR` role and manage this account using a Privileged Account Management (PAM) system. Users who have the `DV_OWNER` role cannot alter this user. Loss of all accounts with the `DV_ACCTMGR` role (such as due to lost passwords or people leaving the organization) is not recoverable. Ensure that a backup `DV_ACCTMGR` account is created for this purpose.
`DV_OWNER` role	Open	Role created during registration and used for managing realms, factors and command rules. This user can add himself or herself to realm authorizations. As a safety measure, maintain a backup user who has the `DV_OWNER` role and manage this account using a Privileged Account Management (PAM) system. Users who have the `DV_OWNER` role cannot alter this user. Loss of all accounts with the `DV_OWNER` role (such as due to lost passwords or people leaving the organization) is not recoverable. Ensure that a backup `DV_OWNER` account is created for this purpose.
`SYSDBA` privilege	Enabled	Privilege created during Oracle Database installation. Required by some Oracle features.
`SYSOPER` privilege	Enabled	Privilege created during Oracle Database installation. Database startup and shutdown. Granted to `SYS` only by default.

Related Topics

- Backup Oracle Database Vault Accounts
  As a best practice, you should maintain backup accounts for the `DV_OWNER` and `DV_ACCTMGR` roles.
- Management of SYSDBA Access
  You should avoid using the `SYS` account and the `SYSDBA` privilege for normal database maintenance tasks.

- Management of SYSOPER Access
  By default, Oracle Database limits `SYSOPER` access to operating system users in the `OSOPER` group and to the user `SYS`.

D.4 Accounts and Roles That Should be Limited to Trusted Individuals

You should limit powerful accounts and roles only to trusted individuals.

- Management of Users with Root Access to the Operating System
  Users who have root user access have full control over the system.
- Management of the Oracle Software Owner
  Users who have access to a system as the Oracle software owner have control over the Oracle software.
- Management of SYSDBA Access
  You should avoid using the `SYS` account and the `SYSDBA` privilege for normal database maintenance tasks.
- Management of SYSOPER Access
  By default, Oracle Database limits `SYSOPER` access to operating system users in the `OSOPER` group and to the user `SYS`.

D.4.1 Management of Users with Root Access to the Operating System

Users who have root user access have full control over the system.

Activities that these users can perform include the following:

- Reading unencrypted files
- Moving and deleting any files
- Starting or stopping any program on the system
- Logging in as any user, including the user who owns the Oracle Database installation

Oracle Database Vault does not provide protection against the operating system root access. Manage the `root` and `oracle` accounts in a Privileged Account Management (PAM) system. Only check these accounts out when they are required for certain tasks. Enhance audit levels when highly privileged operating system accounts are being used, up to an including keystroke capture and video capture.

D.4.2 Management of the Oracle Software Owner

Users who have access to a system as the Oracle software owner have control over the Oracle software.

Activities these users can perform include the following:

- Reading unencrypted database files
- Moving and deleting database files

- Starting or stopping Oracle programs in the system

Oracle Database Vault does not provide protection against the operating system access of the Oracle software owner. Manage the Oracle software owner account in a Privileged Account Management (PAM) system. Only check this account out when it is required for certain tasks. Enhance audit levels when highly privileged operating system accounts are being used, up to an including keystroke capture and video capture.

D.4.3 Management of SYSDBA Access

You should avoid using the `SYS` account and the `SYSDBA` privilege for normal database maintenance tasks.

Instead, use named accounts that have the required system privileges or a specific administrative privilege such as `SYSBACKUP`, `SYSDG`, or `SYSKM`. However, there are cases where the `SYSDBA` privilege is required to perform a patch, upgrade of the database or troubleshoot issues (for example, connecting to a down database).

Because users with the `SYSDBA` privilege could have access to sensitive application data either directly or indirectly (for example, through diagnostics, database upgrades, and patching), use of the `SYSDBA` privilege and accounts must be highly restricted. The list of highly privileged accounts include `SYS` and user accounts with the `SYSDBA` privilege in the database, and the root and oracle accounts in the operating system. Access to highly privileged accounts in the database and the operating system should be on an exception basis and require the user to go through a process to unlock access to these accounts and privileges. Oracle recommends that you manage these accounts with a Privileged Account Management (PAM) system. Only check these accounts out when they are required for certain tasks. Enhance audit levels when highly privileged operating system accounts (`root` and `oracle`) and database accounts (`SYS` account and `SYSDBA` administrative privilege) are being used, up to an including keystroke capture and video capture. When these highly privileged accounts access the database, audit the `SYS` account to monitor their activities. Oracle recommends that you use the `ENABLE_DV_PATCH_ADMIN_AUDIT` procedure during patching operations when the `DV_PATCH_ADMIN` role is granted to `SYS` (or to users who have the with `SYSDBA` administrative privilege).

Related Topics

- ENABLE_DV_PATCH_ADMIN_AUDIT Procedure
  The `ENABLE_DV_PATCH_ADMIN_AUDIT` procedure enables realm, command rule, and rule set auditing of the actions by users who have the `DV_PATCH_ADMIN` role.

D.4.4 Management of SYSOPER Access

By default, Oracle Database limits `SYSOPER` access to operating system users in the `OSOPER` group and to the user `SYS`.

This prevents `SYSOPER` from modifying the Oracle data dictionary directly. The `SYSOPER` privilege has limited privileges within the database, but individuals with this role can start and shut down the Oracle database. Only grant the `SYSOPER` privilege to trusted individuals.

D.5 Guidelines for Using Oracle Database Vault in a Production Environment

You should follow special guidelines when you run Oracle Database Vault in a production environment.

These guidelines are as follows:

- Run a full test of your applications to ensure that the Database Vault policies you have created are working as expected
- Monitor the performance of your applications, and if necessary, tune your rule expressions
- Assign responsibilities to the appropriate production support and security groups, as follows:
  - Assign security responsibilities to the database security administrator.
  - Assign account management to the database account manager.
  - Assign resource management tasks to database administrators.
- Back up your Database Vault API scripts to a secure server.

D.6 Secure Configuration Guidelines

You should be aware of security considerations for special PL/SQL packages, privileges, and the recycle bin.

- General Secure Configuration Guidelines
  General secure configuration guidelines involved patches and revoke operations.
- UTL_FILE and DBMS_FILE_TRANSFER Package Security Considerations
  You should carefully restrict access to the `UTL_FILE` and `DBMS_FILE_TRANSFER` PL/SQL packages.
- CREATE ANY JOB Privilege Security Considerations
  The `CREATE ANY JOB` privilege has been revoked from the `DBA` and the `SCHEDULER_ADMIN` roles.
- CREATE EXTERNAL JOB Privilege Security Considerations
  The `CREATE EXTERNAL JOB` privilege was introduced in Oracle Database 10*g* release 2 (10.2).
- ALTER SYSTEM and ALTER SESSION Privilege Security Considerations
  You should be aware of ways to secure the powerful `ALTER SYSTEM` and `ALTER SESSION` system privileges.

D.6.1 General Secure Configuration Guidelines

General secure configuration guidelines involved patches and revoke operations.

- Installing patches and new applications might re-grant some of the privileges that Oracle recommends that you revoke in this section. Check these privileges after you install patches and new applications to verify that they are still revoked.

- When you revoke `EXECUTE` privileges on packages, ensure that you grant `EXECUTE` on the packages to the owner, check the package dependencies, and recompile any invalid packages after the revoke.

  To find users who have access to the package, log into the database instance as a named database administrator and issue the following query.

  ```
  SELECT * FROM DBA_TAB_PRIVS WHERE TABLE_NAME = package_name;
  ```

  package_name is the name of the package you are looking for.

  To find the users, packages, procedures, and functions that are dependent on the package, issue this query:

  ```
  SELECT OWNER, NAME, TYPE  FROM ALL_DEPENDENCIES
  WHERE REFERENCED_NAME = package_name;
  ```

  Note that these two queries do not identify references to packages made through dynamic SQL.

D.6.2 UTL_FILE and DBMS_FILE_TRANSFER Package Security Considerations

You should carefully restrict access to the `UTL_FILE` and `DBMS_FILE_TRANSFER` PL/SQL packages.

- About Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages
  The `UTL_FILE` package is owned by `SYS` and granted to `PUBLIC`.
- Securing Access to the DBMS_FILE_TRANSFER Package
  You can secure access to the `DBMS_FILE_TRANSFER` PL/SQLpackage in a variety of ways.
- Example: Creating a Command Rule to Deny Access to CREATE DATABASE LINK
  The `DBMS_MACADM.CREATE_COMMAND_RULE` enables you to create command rules to deny access to the `CREATE DATABASE LINK` SQL statement.
- Example: Creating a Command Rule to Enable Access to CREATE DATABASE LINK
  The `DBMS_MACADM.UPDATE_COMMAND_RULE` procedure can be used to modify an existing command rule.
- Example: Command Rules to Disable and Enable Access to CREATE DIRECTORY

D.6.2.1 About Security Considerations for the UTL_FILE and DBMS_FILE_TRANSFER Packages

The `UTL_FILE` package is owned by `SYS` and granted to `PUBLIC`.

However, a user must have access to the directory object to manipulate the files in that operating system directory.

The `DBMS_FILE_TRANSFER` package is owned by `SYS` and granted to the `EXECUTE_CATALOG_ROLE`. Users with `EXECUTE` access on this package can move files from one location to another on the same file system. They also can move files between database instances, including databases on remote systems.

Related Topics

- *Oracle Database PL/SQL Packages and Types Reference*

D.6.2.2 Securing Access to the DBMS_FILE_TRANSFER Package

You can secure access to the `DBMS_FILE_TRANSFER` PL/SQLpackage in a variety of ways.

- Use any of the following methods to secure the `DBMS_FILE_TRANSFER` PL/SQLpackage:
  - Revoke the `EXECUTE` privilege from the `DBMS_FILE_TRANSFER` package and grant the `EXECUTE` privilege only to trusted users who need it.
  - Create command rules to control the `CREATE DATABASE LINK` and `CREATE DIRECTORY` SQL statements. See Creating a Command Rule for information on creating command rules by using Oracle Database Vault Administrator.
  - Create Oracle Database Vault command rules to limit and enable access to the `CREATE DATABASE LINK` and `CREATE DIRECTORY` statements, which are used to establish connections to remote databases.

> **See Also:**
>
> The following sections for examples of command rules that you can create to protect use of the `CREATE DATABASE LINK` statement:
>
> - Example: Creating a Command Rule to Deny Access to CREATE DATABASE LINK
> - Example: Creating a Command Rule to Enable Access to CREATE DATABASE LINK
> - Example: Command Rules to Disable and Enable Access to CREATE DIRECTORY

D.6.2.3 Example: Creating a Command Rule to Deny Access to CREATE DATABASE LINK

The `DBMS_MACADM.CREATE_COMMAND_RULE` enables you to create command rules to deny access to the `CREATE DATABASE LINK` SQL statement.

Example D-1 shows how to create a command rule to deny access to the `CREATE DATABASE LINK` privilege.

Example D-1 Creating a Command Rule to Deny Access to CREATE DATABASE LINK

```
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE (
  command       => 'CREATE DATABASE LINK',
  rule_set_name => 'Disabled',
  object_owner  => '%',
  object_name   => '%',
  enabled       => DBMS_MACUTL.G_YES);
```

```
  END;
  /
COMMIT;
```

D.6.2.4 Example: Creating a Command Rule to Enable Access to CREATE DATABASE LINK

The `DBMS_MACADM.UPDATE_COMMAND_RULE` procedure can be used to modify an existing command rule.

Example D-2 shows how to create a command rule that enables access to the `CREATE DATABASE LINK` privilege.

When a valid user must use the `CREATE DATABASE LINK` statement, the Oracle Database Vault owner can reenable it from Oracle Database Vault Administrator or issue the following commands in SQL*Plus.

Example D-2 Creating a Command Rule to Enable Access to CREATE DATABASE LINK

```
BEGIN
 DBMS_MACADM.UPDATE_COMMAND_RULE (
  command       => 'CREATE DATABASE LINK',
  rule_set_name => 'Enabled',
  object_owner  => '%',
  object_name   => '%',
  enabled       => DBMS_MACUTL.G_YES);
 END;
 /
COMMIT;
```

D.6.2.5 Example: Command Rules to Disable and Enable Access to CREATE DIRECTORY

Example D-3 shows command rules that disable and enable access to `CREATE DIRECTORY`.

Example D-3 Command Rules to Disable and Enable Access to CREATE DIRECTORY

```
-- Disable access to CREATE DIRECTORY
BEGIN
 DBMS_MACADM.CREATE_COMMAND_RULE (
  command       => 'CREATE DIRECTORY',
  rule_set_name => 'Disabled',
  object_owner  => '%',
  object_name   => '%',
  enabled       => dbms_macutl.g_yes);
 END;
  /
COMMIT;

-- Enable access to CREATE DIRECTORY
BEGIN
 dbms_macadm.update_command_rule (
  command       => 'CREATE DIRECTORY',
  rule_set_name => 'Enabled',
  object_owner  => '%',
  object_name   => '%',
  enabled       => dbms_macutl.g_yes);
```

```
 END;
 /
COMMIT;
```

D.6.3 CREATE ANY JOB Privilege Security Considerations

The `CREATE ANY JOB` privilege has been revoked from the `DBA` and the `SCHEDULER_ADMIN` roles.

Ensure that this change does not affect your applications.

Related Topics

- Using Oracle Scheduler with Oracle Database Vault
  Users who are responsible for scheduling database jobs must have Oracle Database Vault-specific authorization.

D.6.4 CREATE EXTERNAL JOB Privilege Security Considerations

The `CREATE EXTERNAL JOB` privilege was introduced in Oracle Database 10*g* release 2 (10.2).

This privilege is required for database users who want to run jobs that run on the operating system outside the database. By default, the `CREATE EXTERNAL JOB` privilege is granted to all users who have been granted the `CREATE JOB` privilege. For greater security, revoke this privilege from users who do not need it and then grant it only to those users who do need it.

D.6.5 ALTER SYSTEM and ALTER SESSION Privilege Security Considerations

You should be aware of ways to secure the powerful `ALTER SYSTEM` and `ALTER SESSION` system privileges.

- About ALTER SYSTEM and ALTER SESSION Privilege Security Considerations
  Be aware that trace and debug commands have the potential to show Oracle database memory information.
- Example: Adding Rules to the Existing ALTER SYSTEM Command Rule
  You can create a rule that prevents users with the `ALTER SYSTEM` privilege from issuing `ALTER SYSTEM` statements.

D.6.5.1 About ALTER SYSTEM and ALTER SESSION Privilege Security Considerations

Be aware that trace and debug commands have the potential to show Oracle database memory information.

Oracle Database Vault does not protect against these commands. To help secure the Oracle database memory information, Oracle recommends that you strictly control access to the `ALTER SYSTEM` and `ALTER SESSION` privileges. These privileges can be granted by the user `SYS` when connected as `SYSDBA` and by any user granted the `DBA` role.

Oracle also recommends that you add rules to the existing command rule for `ALTER SYSTEM` statement. You can use Oracle Database Vault Administrator to create a rule and add it to a rule set. You should grant the `ALTER SESSION` privilege only to trusted users. (For example, the `ALTER SESSION` statement can enable tracing.)

D.6.5.2 Example: Adding Rules to the Existing ALTER SYSTEM Command Rule

You can create a rule that prevents users with the `ALTER SYSTEM` privilege from issuing `ALTER SYSTEM` statements.

Example D-4 shows how to create a rule that prevents users with `ALTER SYSTEM` privilege from issuing the `ALTER SYSTEM DUMP` statement. Log into the database instance as the Oracle Database Vault Owner when you create this command rule.

Alternatively, you can use Oracle Database Vault Administrator to create and add this rule to the rule set. See Creating a Rule to Add to a Rule Set for more information.

Example D-4 Adding Rules to the Existing ALTER SYSTEM Command Rule

```
CONNECT accts_admin_ace
Enter password: password

BEGIN
 DBMS_MACADM.CREATE_RULE('NO_SYSTEM_DUMP',
 '(INSTR(UPPER(DV_SQL_TEXT),''DUMP'') = 0)');
 END;
/
EXEC DBMS_MACADM.ADD_RULE_TO_RULE_SET
  ('Allow Fine Grained Control of System Parameters','NO_SYSTEM_DUMP');

COMMIT;
```

E

Troubleshooting Oracle Database Vault

You can troubleshoot Oracle Database Vault by using tools such as trace files or checking certain Oracle Database Vault reports.

- Using Trace Files to Diagnose Oracle Database Vault Events
  Trace files, which the database generates, capture important information to help you debug errors.
- General Diagnostic Tips
  Oracle provides general tips for diagnosing problems in realms, factors, and rule sets.
- Configuration Problems with Oracle Database Vault Components
  Oracle Database Vault provides reports to check configuration problems with realms, command rules, factors, rule sets, or secure application roles.
- Resetting Oracle Database Vault Account Passwords
  Backup accounts can help you reset lost passwords for users who have been granted the `DV_OWNER` and `DV_ACCTMGR` roles.

E.1 Using Trace Files to Diagnose Oracle Database Vault Events

Trace files, which the database generates, capture important information to help you debug errors.

- About Using Trace Files to Diagnose Oracle Database Vault Events
  You can monitor the Oracle Database Vault database instance for server and background process events by enabling and checking the database instance trace files.
- Types of Oracle Database Vault Trace Events That You Can and Cannot Track
  You can use trace files to track a variety of Oracle Database Vault activities.
- Levels of Oracle Database Vault Trace Events
  You can use the several levels for Oracle Database Vault trace events.
- Performance Effect of Enabling Oracle Database Vault Trace Files
  Be careful about enabling trace files.
- Enabling Oracle Database Vault Trace Events
  You can use the `ALTER SESSION` or `ALTER SYSTEM` SQL statements to enable Oracle Database Vault trace events.
- Finding Oracle Database Vault Trace File Data
  The Linux `grep` command and the ADR Command Interpreter (`ADRCI`) command-line utility can find Oracle Database Vault trace file data.
- Disabling Oracle Database Vault Trace Events
  You can disable tracing for Oracle Database Vault events.

E.1.1 About Using Trace Files to Diagnose Oracle Database Vault Events

You can monitor the Oracle Database Vault database instance for server and background process events by enabling and checking the database instance trace files.

Trace files reveal the Oracle Database Vault policy authorization success and failures. They are useful for providing information to help resolve bug and other issues that may occur.

To set tracing for Oracle Database Vault, you must have the `DV_ADMIN` role. To perform the configuration, you use either of the `ALTER SESSION SET EVENTS` or `ALTER SYSTEM SET EVENTS` SQL statements.

Related Topics

- *Oracle Database Administrator's Guide*

E.1.2 Types of Oracle Database Vault Trace Events That You Can and Cannot Track

You can use trace files to track a variety of Oracle Database Vault activities.

Table E-1 describes these activities.

Table E-1 Contents of Oracle Database Vault Trace Files

Database Vault Feature	Description
Realm authorizations	The trace file tracks cases of realm authorization with a rule set and realm authorization to a role.
Rule set evaluations	The trace file includes information about a rule set evaluation from a realm authorization, for a command rule, the CONNECT command rule, and from a factor.
Oracle Data Pump authorization	The trace file includes Database Vault Data Pump authorization results and other user, object, and SQL text information.
Oracle Scheduler job authorization	The trace file includes the Database Vault Oracle Scheduler job authorization results, job name, job owner, current statement, and so on.
Object privilege bypass	The trace file tracks both direct grants and grants through a role. This type of trace is useful for cases where mandatory realms are not enabled, which enables users who have an object privilege to access realm protected objects.
Factor loading	The trace file tracks the expression and value for each factor loaded.
Others	Object owner bypassed realm protection and other Database Vault failed and succeeded operations

Related Topics

- Example: Low Level Oracle Database Vault Realm Violations in a Trace File
  You can use trace file data to track low level realm violations.

E.1.3 Levels of Oracle Database Vault Trace Events

You can use the several levels for Oracle Database Vault trace events.

These levels are as follows:

- **Low** prints the information for all failed Oracle Database Vault authorizations to a trace file. This type of trace file includes failed realm authorizations, failed factor loading, failed rule set evaluating, and so on. It has a low impact on Oracle Database performance.
- **High** prints trace records that include both successful and failed authorizations. Because this type of tracing tracks all the authorizations, the overhead is larger than that of the low level tracing. In addition, the trace files are usually larger.
- **Highest** prints the PL/SQL stack and function call stack to a trace file, as well as what is traced at level high (as described in Table E-1). It has the highest impact on Oracle Database performance.

E.1.4 Performance Effect of Enabling Oracle Database Vault Trace Files

Be careful about enabling trace files.

Doing so can increase the overhead of the database instance operation, which could decrease performance.

E.1.5 Enabling Oracle Database Vault Trace Events

You can use the `ALTER SESSION` or `ALTER SYSTEM` SQL statements to enable Oracle Database Vault trace events.

- Enabling Trace Events for the Current Database Session
  You can use the `ALTER SESSION SET EVENTS` SQL statement to enable trace events for the current database session.
- Enabling Trace Events for All Database Sessions
  You can use the `ALTER SYSTEM SET EVENTS` SQL statement to enable Database Vault trace events for all database sessions.
- Enabling Trace Events in a Multitenant Environment
  Trace events affect both the current user session and all database sessions.

E.1.5.1 Enabling Trace Events for the Current Database Session

You can use the `ALTER SESSION SET EVENTS` SQL statement to enable trace events for the current database session.

1. Log into the database instance as a user who has been granted the `DV_ADMIN` role and the `ALTER SESSION` system privilege.
2. Enter the `ALTER SESSION SET EVENTS` SQL statement to set the level of the Oracle Database Vault trace events to low, high, or highest.
   - To turn on tracing for failed operations that have a low impact, enter one of the following statements:

```
ALTER SESSION SET EVENTS 'TRACE[DV] DISK=LOW';

ALTER SESSION SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 1';
```

- To turn on tracing for both failed and successful operations that have a high impact, enter one of the following statements:

```
ALTER SESSION SET EVENTS 'TRACE[DV] DISK=HIGH';

ALTER SESSION SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 3';
```

- To turn on tracing for both failed and successful operations with a function and PL/SQL call stack that has the highest impact, enter one of the following statements:

```
ALTER SESSION SET EVENTS 'TRACE[DV] DISK=HIGHEST';

ALTER SESSION SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 4';
```

Related Topics

- Levels of Oracle Database Vault Trace Events
  You can use the several levels for Oracle Database Vault trace events.

E.1.5.2 Enabling Trace Events for All Database Sessions

You can use the `ALTER SYSTEM SET EVENTS` SQL statement to enable Database Vault trace events for all database sessions.

Enabling Oracle Database Vault trace events generates trace files that consume disk space. Oracle recommends that you only enable trace events during the period of activity that you need to capture.

1. Log into the database instance as a user who has been granted the `DV_ADMIN` role and the `ALTER SYSTEM` system privilege.
2. Enter the `ALTER SYSTEM SET EVENTS` SQL statement to set the level of the Oracle Database Vault trace events to low, high, or highest.
   - To turn on tracing for failed operations that have a low impact, enter one of the following statements:

```
ALTER SYSTEM SET EVENTS 'TRACE[DV] DISK=LOW';

ALTER SYSTEM SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 1';
```

   - To turn on tracing for both failed and successful operations that have a high impact, enter one of the following statements:

```
ALTER SYSTEM SET EVENTS 'TRACE[DV] DISK=HIGH';

ALTER SYSTEM SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 3';
```

   - To turn on tracing for both failed and successful operations with a function and PL/SQL call stack that has the highest impact, enter one of the following statements:

```
ALTER SYSTEM SET EVENTS 'TRACE[DV] DISK=HIGHEST';

ALTER SYSTEM SET EVENTS '47998 TRACE NAME CONTEXT FOREVER, LEVEL 4';
```

Another way that you can enable trace events for all database sessions is to add the following line to the `init.ora` file, and then restart the database:

```
event="47998 trace name context forever, level [trace_level]"
```

Replace *trace_level* with one of the following values:

- 1 for the lowest level of tracing
- 3 for the high level
- 4 for the highest level

For example:

```
event="47998 trace name context forever, level [1]"
```

Related Topics

- Disabling Trace Events for All Database Sessions
  You can use the ALTER SYSTEM SET EVENTS SQL statement to disable Database Vault tracing for all database sessions.
- Levels of Oracle Database Vault Trace Events
  You can use the several levels for Oracle Database Vault trace events.

E.1.5.3 Enabling Trace Events in a Multitenant Environment

Trace events affect both the current user session and all database sessions.

- **Trace events for the current user session:** Running the ALTER SESSION SET EVENTS SQL statement from either the root or a pluggable database (PDB) enables tracing for the current user session. If you switch from one PDB to another PDB (by using the ALTER SESSION SET CONTAINER statement), then tracing is still enabled for the new PDB. You cannot enable tracing for a single PDB; the tracing applies to all PDBs and the root. Remember that you must have the ALTER SESSION SET CONTAINER system privilege to move from one PDB to another.
- **Trace events for all database sessions:** Running the ALTER SYSTEM SET EVENTS statement from either the root or a specific PDB enables tracing for all PDBs in the container database.

E.1.6 Finding Oracle Database Vault Trace File Data

The Linux grep command and the ADR Command Interpreter (ADRCI) command-line utility can find Oracle Database Vault trace file data.

- Finding the Database Vault Trace File Directory Location
  You can find the full directory location of trace files by querying the V$DIAG_INFO dynamic view.
- Using the Linux grep Command to Search Trace Files for Strings
  To query or process the trace files, you can use the Linux grep command to search for strings.
- Using the ADR Command Interpreter (ADRCI) Utility to Query Trace Files
  You can query trace files by using the ADR Command Interpreter (ADRCI) command-line utility.
- Example: Low Level Oracle Database Vault Realm Violations in a Trace File
  You can use trace file data to track low level realm violations.

- Example: High Level Trace Enabled for Oracle Database Vault Authorization
  You can track Oracle Database Vault authorizations in a trace file with high level trace enabled.
- Example: Highest Level Traces on Violations on Realm-Protected Objects
  You can track high level violations using trace files.

E.1.6.1 Finding the Database Vault Trace File Directory Location

You can find the full directory location of trace files by querying the `V$DIAG_INFO` dynamic view.

- Query the `V$DIAG_INFO` dynamic view as follows:

```
SELECT VALUE FROM V$DIAG_INFO WHERE NAME = 'Default Trace File';
```

Output similar to the following appears:

```
VALUE
--------------------------------------------------------------------------------
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_7174.trc
```

E.1.6.2 Using the Linux grep Command to Search Trace Files for Strings

To query or process the trace files, you can use the Linux `grep` command to search for strings.

- For example, to find the trace files that show realm authorization failures, enter the following command:

```
grep 'Result=Realm Authorization Failed'  *.trc
```

E.1.6.3 Using the ADR Command Interpreter (ADRCI) Utility to Query Trace Files

You can query trace files by using the ADR Command Interpreter (`ADRCI`) command-line utility.

- To use the `ADRCI` utility to find trace file information, use the `SHOW` command.

  For example, to use `ADRCI` to find the trace files, enter the `SHOW TRACEFILE` command:

```
adrci --To start ACRCI from the command line
adrci> show tracefile

diag/rdbms/orcl/orcl/trace/orcl_m002_14551.trc
diag/rdbms/orcl/orcl/trace/orcl_tmon_13450.trc
diag/rdbms/orcl/orcl/trace/orcl_vktm_963.trc
diag/rdbms/orcl/orcl/trace/alert_orcl.log
...
```

  To find the number of all trace incidents:

```
adrci> show incident

ADR Home = /u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl:
*************************************************************************
234 rows fetched
```

The following `ADRCI` command returns a list of all trace files whose name contains the word `ora`:

```
adrci> show tracefile %ora%

/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_18841.trc
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_12017.trc
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_19372.trc
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_12221.trc
/u01/app/oracle/product/12.1.0/log/diag/rdbms/orcl/orcl/trace/orcl_ora_1600.trc
...
```

The following `ADRCI` command searches for trace files that contain the phrase `Realm Authorization Failed`:

```
adrci> show trace %trc -xp "[payload like '%Realm Authorization Failed%']"
```

Related Topics

- *Oracle Database Utilities*
- *Oracle Database Administrator's Guide*

E.1.6.4 Example: Low Level Oracle Database Vault Realm Violations in a Trace File

You can use trace file data to track low level realm violations.

Example E-1 shows an example of tracking low lever real violations.

Example E-1 Low Level Oracle Database Vault Realm Violations in a Trace File

```
*** 2010-02-05 18:35:31.438
*** SESSION ID:(34.559) 2010-02-05 18:35:31.438
*** CLIENT ID:() 2010-02-05 18:35:31.438
*** SERVICE NAME:(SYS$USERS) 2010-02-05 18:35:31.438
*** MODULE NAME:(SQL*Plus) 2010-02-05 18:35:31.438
*** ACTION NAME:() 2010-02-05 18:35:31.438

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=INSERT INTO U1.T1 VALUES(30)

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=DELETE FROM U1.T1

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T3  Object_Type=TABLE
        SQL_Text=CREATE TABLE U1.T3(C INT)

*** 2010-02-05 18:35:34.465

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
```

```
        SQL_Text=INSERT INTO U1.T1 VALUES(30)

Result=Realm Authorization Failed
        Realm_Name=realm 3      Required_Auth_Level=0
        Current_User=116
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=DELETE FROM U1.T1
```

E.1.6.5 Example: High Level Trace Enabled for Oracle Database Vault Authorization

You can track Oracle Database Vault authorizations in a trace file with high level trace enabled.

Example E-2 shows an example of this type of trace file.

Example E-2 High Level Trace Enabled for Oracle Database Vault Authorization

```
Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$AUDIT  Object_Type=TABLE

Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$AUDIT  Object_Type=TABLE

Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$POL    Object_Type=TABLE

Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$USER_LOGON     Object_Type=VIEW

……

Result= Realm Authorization Passed
        Reason=Current user is the object owner
        Current_User=70 Command=SELECT
        Object_Owner=LBACSYS    Object_Name=LBAC$POL    Object_Type=TABLE

Result=Set Factor Value
        Factor_Name=Sensitive_Treatments        Factor_Expression=/SURGERY/
PSYCHOLOGICAL

Result=Set Factor Value
        Factor_Name=Database_Instance
Factor_Expression=UPPER(SYS_CONTEXT('USERENV','INSTANCE'))      Factor_Value=1

Result=Set Factor Value
        Factor_Name=Client_IP
Factor_Expression=UPPER(SYS_CONTEXT('USERENV','IP_ADDRESS'))    Factor_Value=

Result=Set Factor Value
        Factor_Name=Authentication_Method
Factor_Expression=UPPER(SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD'))
```

```
Factor_Value=PASSWORD
……

*** ACTION NAME:() 2010-02-05 18:47:19.540

Result=Rule Set Evaluation Failed
        Command=SELECT  RuleSet_ID=2    RuleSet_Name=Disabled
        Current_User=SYSTEM
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=SELECT * FROM U1.T1

Result=Rule Set Evaluation Succeeded
        Command=SELECT  RuleSet_ID=1    RuleSet_Name=Enabled
        Current_User=SYSTEM
        Object_Owner=U1 Object_Name=T1  Object_Type=TABLE
        SQL_Text=SELECT * FROM U1.T1
```

E.1.6.6 Example: Highest Level Traces on Violations on Realm-Protected Objects

You can track high level violations using trace files.

Example E-3 shows how highest level violations that involve Oracle Scheduler jobs authorization can appear in a trace file when trace is enabled at the highest level.

Example E-3 Highest Level Traces on Violations on Realm-Protected Objects

```
------ Call Stack Trace ------
kzvdvechk<-kzvdveqau<-kksfbc<-opiexe<-kpoal8<-opiodr<-ttcpip<-opitsk<-opiino<-opiodr<-
opidrv<-sou2o<-opimai_real<-ssthrdmain<-main<-__libc_start_main<-_start

Result=Object Privilege check passed
        Current_User=INVOKER2   Used_Role=1
        Object_Owner=SYSTEM     Object_Name=PRODUCT_PRIVS       Object_Type=VIEW
        SQL_Text=SELECT CHAR_VALUE FROM SYSTEM.PRODUCT_PRIVS WHERE (UPPER('SQL*PLUS')
LIKE UPPER(PRODUCT)) AND ((USER LIKE USERID) OR (USERID = 'PUBLIC')) AND
(UPPER(ATTRIBUTE) = 'ROLES')
*** MODULE NAME:(SQL*Plus) 2010-02-05 18:57:53.973
*** ACTION NAME:() 2010-02-05 18:57:53.973

----- Current SQL Statement for this session (sql_id=2sr63rjm45yfh) -----
UPDATE INVOKER1.T1 SET A = 20
----- PL/SQL Stack -----
----- PL/SQL Call Stack -----
  object      line  object
  handle    number  name
0x26a00e34         1  anonymous block
0x2495b000       185  package body SYS.DBMS_ISCHED
0x24958fb8       486  package body SYS.DBMS_SCHEDULER
0x247bbb34         1  anonymous block

------ Call Stack Trace ------
kzvdvechk<-kzvdveqau<-kksfbc<-opiexe<-opipls<-opiodr<-__PGOSF151_rpidrus<-skgmstack<-
rpidru<-rpiswu2<-rpidrv<-psddr0<-psdnal<-pevm_EXECC<-pfrinstr_EXECC<-pfrrun_no_tool<-
pfrrun<-plsql_run<-peicnt<-kkxexe<-opiexe<-kpoal8<-opiodr<-kpoodr<-upirtrc<-kpurcsc<-
kpuexec
<-OCIStmtExecute<-jslvec_execcb<-jslvswu<-jslve_execute0<-jskaJobRun<-jsiRunJob<-
jsaRunJob<-spefcmpa<-spefmccallstd<-pextproc<-__PGOSF495_peftrusted<-
__PGOSF522_psdexsp<-rpiswu2<-psdextp<-pefccal<-pefcal<-pevm_FCAL<-pfrinstr_FCAL<-
pfrrun_no_tool<-pfrrun<-plsql_run
<-peicnt<-kkxexe<-opiexe<-kpoal8<-opiodr<-ttcpip<-opitsk<-opiino<-opiodr<-opidrv<-
sou2o<-opimai_real<-ssthrdmain<-main<-__libc_start_main<-_start
```

```
Result=Realm Authorization Succeeded
        Realm_Name=jobowner realm       Used_Auth_Level=0
        Current_User=119
        Object_Owner=INVOKER1   Object_Name=T1  Object_Type=TABLE
        SQL_Text=UPDATE INVOKER1.T1 SET A = 20

Result=Scheduler Job Authorization Succeeded
        Current_User=JOBOWNER   Logon_User=INVOKER2
        Job_Owner=JOBOWNER      Job_Name=DMLJOB1
        Object_Owner=INVOKER1   Object_Name=T1  Object_Type=TABLE
        SQL_Text=UPDATE INVOKER1.T1 SET A = 20
```

E.1.7 Disabling Oracle Database Vault Trace Events

You can disable tracing for Oracle Database Vault events.

- Disabling Trace Events for the Current Database Session
  You can use the `ALTER SESSION SET EVENTS` SQL statement to disable Database Vault tracing for the current database session.
- Disabling Trace Events for All Database Sessions
  You can use the `ALTER SYSTEM SET EVENTS` SQL statement to disable Database Vault tracing for all database sessions.
- Disabling Trace Events in a Multitenant Environment
  Disabling trace events affects both the current user session and all databaes sessions.

E.1.7.1 Disabling Trace Events for the Current Database Session

You can use the `ALTER SESSION SET EVENTS` SQL statement to disable Database Vault tracing for the current database session.

1. Log into the database instance as a user who has been granted the `DV_ADMIN` role and the `ALTER SESSION` system privilege.
2. Enter both of the following SQL statements to disable tracing:

```
ALTER SESSION SET EVENTS 'TRACE[DV] OFF';
ALTER SESSION SET EVENTS '47998 trace name context off';
```

E.1.7.2 Disabling Trace Events for All Database Sessions

You can use the `ALTER SYSTEM SET EVENTS` SQL statement to disable Database Vault tracing for all database sessions.

1. Log into the database instance as a user who has been granted the `DV_ADMIN` role and the `ALTER SYSTEM` system privilege.
2. Enter the following `ALTER SYSTEM SET EVENTS` SQL statements.

```
ALTER SYSTEM SET EVENTS 'TRACE[DV] OFF';
ALTER SYSTEM SET EVENTS '47998 trace name context off';
```

Another way that you can disable trace events for all database sessions is to add the following line to the `init.ora` file, and then restart the database:

```
event="47998 trace name context off"
```

Ensure that the `init.ora` file does not have any conflicting `47998` lines, such as `event="47998 trace name context forever, level [1]"`.

E.1.7.3 Disabling Trace Events in a Multitenant Environment

Disabling trace events affects both the current user session and all databaes sessions.

- **Trace events for the current user session:** Running the `ALTER SESSION SET EVENTS` SQL statement from either the root or a PDB disables tracing for the current user session. If you switch from one PDB to another PDB (by using the `ALTER SESSION SET CONTAINER` statement), then tracing is still disabled for the new PDB. You cannot disable tracing for a single PDB; the tracing applies to all PDBs and the root. Remember that you must have the `ALTER SESSION SET CONTAINER` system privilege to move from one PDB to another.
- **Trace events for all database sessions:** Running the `ALTER SYSTEM SET EVENTS` statement from either the root or a specific PDB disables tracing for all PDBs in the CDB.

E.2 General Diagnostic Tips

Oracle provides general tips for diagnosing problems in realms, factors, and rule sets.

These guidelines are as follows:

- For realm protections, verify that a user has the underlying system or object privileges (granted directly or through a role) that might affect the command.
- If a realm authorization is not working, verify that the account roles are set correctly.
- For PL/SQL expressions used in factors and rule sets, grant the `EXECUTE` privilege on the PL/SQL package functions used in these expressions directly to the account and determine if the results appear to be correct.
- Use the auditing reports to diagnose problems in general.

Related Topics

- Oracle Database Vault Reports
  Oracle Enterprise Manager provides Oracle Database Vault-related reports.

E.3 Configuration Problems with Oracle Database Vault Components

Oracle Database Vault provides reports to check configuration problems with realms, command rules, factors, rule sets, or secure application roles.

See the following sections for more information:

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

- Secure Application Configuration Issues Report

To run these reports, see Running the Oracle Database Vault Reports.

E.4 Resetting Oracle Database Vault Account Passwords

Backup accounts can help you reset lost passwords for users who have been granted the DV_OWNER and DV_ACCTMGR roles.

- Resetting the DV_OWNER User Password
  You can use the DV_OWNER backup account to reset the DV_OWNER user password.
- Resetting the DV_ACCTMGR User Password
  You can use the DV_ACCTMGR backup account to reset the DV_ACCTMGR user password.

E.4.1 Resetting the DV_OWNER User Password

You can use the DV_OWNER backup account to reset the DV_OWNER user password.

To reset the DV_OWNER user password, you must temporarily revoke the DV_OWNER role from this user, reset the password, and then re-grant the role back to the user.

1. Log in to the database instance as the backup user for the DV_OWNER user account.
2. Revoke the DV_OWNER role from the DV_OWNER user who has lost the password.

   For example:

   ```
   REVOKE DV_OWNER FROM sec_admin_owen;
   ```

3. Connect as a user who has been granted the DV_ACCTMGR role.
4. Reset the password for the DV_OWNER user.

   ```
   ALTER USER sec_admin_owen IDENTIFIED BY password;
   ```

   Replace *password* with a password that meets the password complexity requirements of the user's profile.
5. Connect as the backup DV_OWNER user.
6. Grant the DV_OWNER role back to the DV_OWNER user.

   ```
   GRANT DV_OWNER TO sec_admin_owen WITH ADMIN OPTION;
   ```

> **Note:**
>
> Ensure that the backup DV_OWNER account is safely stored in case it is needed again.

Related Topics

- *Oracle Database Security Guide*

E.4.2 Resetting the DV_ACCTMGR User Password

You can use the `DV_ACCTMGR` backup account to reset the `DV_ACCTMGR` user password.

To reset the `DV_ACCTMGR` user password, you can use the backup `DV_ACCTMGR` account to reset this user's password.

1. Log in to the database instance as the backup user for the `DV_ACCTMGR` user account.
2. Reset the password for the `DV_ACCTMGR` user.

   For example:

   ```
   ALTER USER accts_admin_ace IDENTIFIED BY password;
   ```

   Replace *password* with a password that meets the password complexity requirements of the user's profile.

> **Note:**
>
> Ensure that the backup `DV_ACCTMGR` account is safely stored in case it is needed again.

Related Topics

- *Oracle Database Security Guide*

Index

A

access control policy
  reports
    Core Database Vault Audit Report, *26-6*
Access to Sensitive Objects Report, *26-11*
accounts
  See database accounts
Accounts With DBA Roles Report, *26-14*
Accounts with SYSDBA/SYSOPER Privilege Report, *26-12*
ad hoc tools
  preventing use of, *7-23*
administrators
  DBA operations in Oracle Database Vault, *12-1*
ADRCI utility
  Database Vault, *E-6*
alerts
  Enterprise Manager Cloud Control, *12-9*
ALTER ROLE statement
  monitoring, *25-1*
ALTER SESSION command rules, *6-4*, *16-17*
  about, *6-4*
ALTER SESSION event command rules
  creating, *16-11*
  updating, *16-23*
ALTER SESSION privilege
  enabling trace files, *E-3*
  reports, ALTER SYSTEM or ALTER SESSION Report, *26-14*
ALTER SESSION statement
  guidelines on managing privileges, *D-14*
ALTER SYSTEM command rules
  deleting system event command rules, *16-18*
ALTER SYSTEM event command rules
  creating, *16-13*
  updating, *16-24*
ALTER SYSTEM or ALTER SESSION Report, *26-14*
ALTER SYSTEM privilege
  reports, ALTER SYSTEM or ALTER SESSION Report, *26-14*
ALTER SYSTEM statement
  guidelines on managing privileges, *D-14*
ALTER USER statement
  monitoring, *25-1*
ANY System Privileges for Database Accounts Report, *26-10*
audit policy change
  monitoring, *25-1*
AUDIT privilege, *26-16*
AUDIT Privileges Report, *26-16*
AUDIT_TRAIL$ system table
  affected by AUDIT_TRAIL initialization parameter, *A-4*
  archiving, *A-6*
  format, *A-4*
  purging, *A-8*
auditing
  about, *A-1*
  archiving Database Vault audit trail, *A-6*
    about, *A-6*
  Core Database Audit Report, *26-17*
  DBMS_MACUTL fields, *20-1*
  purging Database Vault audit trail, *A-8*
    about, *A-6*
  realms
    DBMS_MACUTL fields, *20-1*
    options, *4-10*
  reports, *26-5*
  rule sets
    DBMS_MACUTL fields, *20-1*
    options, *5-4*
  secure application roles
    audit records, *8-10*
auditing policies
  about, *A-1*
  audit events
    about, *A-3*
  custom events
    audit trail, *A-4*
  events that are tracked, *A-3*
  monitoring changes to, *25-1*
auditing, authorizations for
  about, *12-38*
  DBA_DV_AUDIT_ADMIN_AUTH view, *24-7*
  DBA_DV_AUDIT_VIEWER_AUTH view, *24-7*
  DBMS_MACADM.AUTHORIZE_ADMIN_VIEWER procedure, *21-10*

auditing, authorizations for *(continued)*
DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN procedure, *21-10*
DBMS_MACADM.UNAUTHORIZE_ADMIN_USER procedure, *21-28*
DBMS_MACADM.UNAUTHORIZE_ADMIN_VIEWER procedure, *21-29*
granting AUDIT_ADMIN authorization, *12-39*
granting AUDIT_VIEWER authorization, *12-40*
Oracle Audit realm, *4-10*
revoking AUDIT_ADMIN authorization, *12-39*
revoking AUDIT_VIEWER authorization, *12-41*
AUDSYS.DV$CONFIGURATION_AUDIT view, *24-65*
AUDSYS.DV$ENFORCEMENT_AUDIT view, *24-65*
authentication
Authentication_Method default factor, *7-2*
command rules, *6-2*
method, finding with DVF.F$AUTHENTICATION_METHOD, *17-29*
realm procedures, *14-1*
authorizations
Oracle Data Pump activities, *12-10*
realms, *4-16*
scheduling database jobs, *12-19*
SQL Firewall activities, *12-5*
AUTHORIZE_MAINTENANCE_USER procedure, *21-14*

B

backup accounts, *13-25*
BECOME USER Report, *26-14*
BECOME USER system privilege
about, *26-14*
break-glass accounts
See backup accounts
break-glass protocol, *12-28*

C

catalog-based roles, *26-15*
CDB_DV_STATUS view, *24-5*
CDBs, *1-10*
Database Vault operations control, *12-28*
functionality in Oracle Database Vault, *1-10*
preventing local users from blocking operations, *12-31*
realms, *4-4*
authorizations, *4-16*
rule sets, *5-2*
CDBS
PDB access by infrastructure DBAs, *12-28*
client identifiers
function to return, *17-32*
clients
finding IP address with DVF.F$CLIENT_IP, *17-29*
code groups
retrieving value with DBMS_MACUTL functions, *20-7*
Command Rule Audit Report, *26-5*
Command Rule Configuration Issues Report, *26-3*
command rules, *6-2*, *6-7*, *6-9*
about, *6-2*
creating, *6-9*
data dictionary view, *6-16*
data masking, *12-36*
default command rules, *6-6*
deleting, *6-11*
editing, *6-9*
functions
DBMS_MACUTL (utility), *20-1*
guidelines, *6-15*
how command rules work, *6-12*
modifying, *6-10*
objects
name, *6-9*
owner, *6-9*
performance effect, *6-16*
procedures
DBMS_MACADM (configuration), *16-1*
process flow, *6-12*
propagating configuration to other databases, *12-7*
reports, *6-16*
rule sets
selecting, *6-9*
used with, *6-2*
simulation mode, *10-1*
troubleshooting
with auditing report, *26-5*
tutorial, *6-13*
views, *6-16*, *24-9*
with PDBs, *6-3*
See also rule sets
common objects, preventing local users from blocking operations
about, *12-31*
common objects, preventing local users from blocking operations of
procedure for, *12-32*
common objects, restricting local user access to
DBMS_MACADM.ALLOW_COMMON_OPERATION procedure, *21-6*
finding status of, *24-55*

compliance
Oracle Database Vault addressing, 1-7
computer name
finding with DVF.F$MACHINE, 17-36
Machine default factor, 7-2
configuration
monitoring changes, 25-1
views
AUDSYS.DV$CONFIGURATION_AUDIT, 24-65
DVSYS.DV$CONFIGURATION_AUDIT, 24-46
DVSYS.DV$ENFORCEMENT_AUDIT, 24-50
configuration and enablement
multitenant, about, 3-3
CONFIGURE_DV procedure
about, 21-38
configuring and enabling Database Vault with, 3-6, 3-8
configuring and enabling Oracle Database Vault, 3-1
CONNECT command rules
about, 6-4
example, 6-4
CONNECT events, controlling with command rules, 6-2
core database
troubleshooting with Core Database Vault Audit Report, 26-6
Core Database Audit Report, 26-17
Core Database Vault Audit Trail Report, 26-6
CPU_PER_SESSION resource profile, 26-16
CREATE ANY JOB privilege, D-14
CREATE ANY JOB statement
guidelines on managing privileges, D-14
CREATE EXTERNAL JOB privilege, D-14
CREATE JOB privilege, D-14
CREATE JOB statement
guidelines on managing privileges, D-14
CREATE ROLE statement
monitoring, 25-1
CREATE USER statement
monitoring, 25-1
CTXSYS schema realm protection, 4-7

D

data definition language (DDL)
statement
controlling with command rules, 6-2
Data Definition Language (DDL) statements
Database Vault authorization
DBA_DV_DDL_AUTH view, 24-13
granting, 21-13
Data Definition Language (DDL) statements *(continued)*
Database Vault authorization *(continued)*
revoking, 21-31
Data Dictionary realm
data masking, 12-35
data manipulation language (DML)
statement
checking with DBMS_MACUTL.CHECK_DVSYS_DML_ALLOW function, 20-7
controlling with command rules, 6-2
data masking
about, 12-35
adding users to realms for, 12-36
creating command rule for, 12-36
errors that can appear, 12-35
data Oracle Database Vault recognizes
See factors
Database Account Default Password Report, 26-17
Database Account Status Report, 26-17
database accounts, 4-7
backup DV_OWNER and DV_ACCTMGR, 13-25
counting privileges of, 26-12
DBSNMP
granted DV_MONITOR role, 13-16
DVSYS, 13-23
LBACSYS, 13-23
monitoring, 25-1
reports
Accounts With DBA Roles Report, 26-14
ALTER SYSTEM or ALTER SESSION Report, 26-14
ANY System Privileges for Database Accounts Report, 26-10
AUDIT Privileges Report, 26-16
BECOME USER Report, 26-14
Database Account Default Password Report, 26-17
Database Account Status Report, 26-17
Database Accounts With Catalog Roles Report, 26-15
Direct and Indirect System Privileges By Database Account Report, 26-9
Direct Object Privileges Report, 26-8
Direct System Privileges By Database Account Report, 26-9
Hierarchical System Privileges by Database Account Report, 26-9
Object Access By PUBLIC Report, 26-7
Object Access Not By PUBLIC Report, 26-8
OS Security Vulnerability Privileges, 26-16

database accounts *(continued)*
reports *(continued)*
Password History Access Report, *26-15*
Privileges Distribution By Grantee Report, *26-12*
Privileges Distribution By Grantee, Owner Report, *26-12*
Privileges Distribution By Grantee, Owner, Privilege Report, *26-13*
Roles/Accounts That Have a Given Role Report, *26-15*
Security Policy Exemption Report, *26-14*
WITH ADMIN Privilege Grants Report, *26-14*
WITH GRANT Privileges Report, *26-15*
solution for lockouts, *B-1*
suggested, *13-24*
Database Accounts With Catalog Roles Report, *26-15*
database administrative operations, *12-1*
database domains, Database_Domain default factor, *7-2*
database links
function to return information about, *17-33*
database objects, *13-1*, *24-32*
Oracle Database Vault, *13-1*
reports
Object Dependencies Report, *26-8*
See also objects
database options, installing, *B-1*
database roles
about, *13-4*
counting privileges of, *26-12*
default Oracle Database Vault, *13-4*
DV_ACCTMGR
about, *13-10*
DV_ADMIN, *13-12*
DV_AUDIT_CLEANUP, *13-13*
DV_DATAPUMP_NETWORK_LINK, *13-14*
DV_GOLDENGATE_ADMIN, *13-15*
DV_GOLDENGATE_REDO_ACCESS, *13-15*
DV_MONITOR, *13-16*
DV_OWNER, *13-17*
DV_PATCH_ADMIN, *13-19*
DV_POLICY_OWNER, *13-20*
DV_SECANALYST, *13-21*
DV_XSTREAM_ADMIN, *13-22*
enabled, determining with ROLE_IS_ENABLED, *17-26*
granting Database Vault roles to users, *13-9*
monitoring, *25-1*
Oracle Database Vault, default, *13-4*
reports
Accounts With DBA Roles Report, *26-14*
database roles *(continued)*
reports *(continued)*
ALTER SYSTEM or ALTER SESSION Report, *26-14*
AUDIT Privileges Report, *26-16*
BECOME USER Report, *26-14*
Database Accounts With Catalog Roles Report, *26-15*
OS Security Vulnerability Privileges, *26-16*
Privileges Distribution By Grantee Report, *26-12*
Roles/Accounts That Have a Given Role Report, *26-15*
Security Policy Exemption Report, *26-14*
WITH ADMIN Privilege Grants Report, *26-14*
separation of duty enforcement, *2-3*
database sessions, *7-9*
controlling with Allow Sessions default rule set, *5-3*
factor evaluation, *7-19*
session user name, Proxy_User default factor, *7-2*
Database Vault, *1-2*
MACADM procedure for deleting operations exception, *21-19*
See also Oracle Database Vault
Database Vault Account Management realm, *4-6*
Database Vault command rule protections, *6-2*
Database Vault operations control
adding users and packages to exception list, how works, *12-28*
adding users and packages to exception list, procedure, *12-30*
deleting users and packages from exception list, *12-30*
disabling, *12-31*
enabling, *12-29*
MACADM procedure enabling operations control, *21-23*
MACADM procedure for adding operations exception, *21-4*
MACADM procedure for disabling operations control, *21-20*
Database Vault realm protection, *4-2*
Database Vault realm protections, *4-2*
databases
defined with factors, *7-1*
domain, Domain default factor, *7-2*
event monitoring, *E-2*
grouped schemas
See realms, *4-2*
host names, Database_Hostname default factor, *7-2*

databases *(continued)*
instance, retrieving information with functions, *17-1*
instances
Database_Instance default factor, *7-2*
names, finding with DVF.F$DATABASE_INSTANCE, *17-31*
number, finding with DV_INSTANCE_NUM, *15-16*
IP addresses
Database_IP default factor, *7-2*
retrieving with DVF.F$DATABASE_IP, *17-31*
monitoring events, *E-2*
names
Database_Name default factor, *7-2*
retrieving with DV_DATABASE_NAME, *15-16*
retrieving with DVF.F$DATABASE_NAME, *17-31*
parameters
Security Related Database Parameters Report, *26-16*
roles that do not exist, *26-4*
schema creation, finding with DVF.F$IDENTIFICATION_TYPE, *17-34*
schema creation, Identification_Type default factor, *7-2*
user name, Session_User default factor, *7-2*
DBA role
impact of Oracle Database Vault installation, *2-3*
DBA_DV_APP_EXCEPTION view, *24-6*
DBA_DV_AUDIT_ADMIN_AUTH view, *24-7*
DBA_DV_AUDIT_VIEWER_AUTH view, *24-7*
DBA_DV_CODE view, *24-8*
DBA_DV_COMMAND_RULE view, *6-16*, *24-9*
DBA_DV_DATAPUMP_AUTH view, *24-11*
DBA_DV_DBCAPTURE_AUTH view, *24-12*
DBA_DV_DBREPLAY_AUTH view, *24-13*
DBA_DV_DDL_AUTH view, *24-13*
DBA_DV_DICTIONARY_ACCTS view, *24-14*
DBA_DV_FACTOR view, *24-14*
DBA_DV_FACTOR_LINK, *24-17*
DBA_DV_FACTOR_LINK view, *24-17*
DBA_DV_FACTOR_TYPE view, *24-17*
DBA_DV_IDENTITY view, *24-18*
DBA_DV_IDENTITY_MAP view, *24-19*
DBA_DV_JOB_AUTH view, *24-20*
DBA_DV_MAC_POLICY view, *24-20*
DBA_DV_MAC_POLICY_FACTOR view, *24-21*
DBA_DV_MAINTENANCE_AUTH view, *24-21*
DBA_DV_ORADEBUG view, *24-22*
DBA_DV_PATCH_ADMIN_AUDIT view, *24-22*
DBA_DV_POLICY view, *24-23*
DBA_DV_POLICY_LABEL view, *24-24*
DBA_DV_POLICY_OBJECT view, *24-25*
DBA_DV_POLICY_OWNER view, *24-26*
DBA_DV_PREPROCESSOR_AUTH view, *24-27*
DBA_DV_PROXY_AUTH view, *24-27*
DBA_DV_PUB_PRIVS view, *24-28*
DBA_DV_REALM view, *24-29*
DBA_DV_REALM_AUTH view, *24-31*
DBA_DV_REALM_OBJECT view, *24-32*
DBA_DV_ROLE view, *24-33*
DBA_DV_RULE view, *24-34*
DBA_DV_RULE_SET view, *24-35*
DBA_DV_RULE_SET_RULE view, *24-38*
DBA_DV_SIMULATION_LOG view, *24-39*
DBA_DV_SQL_FIREWALL_AUTH view, *24-43*
DBA_DV_STATUS view, *24-42*
DBA_DV_TTS_AUTH view, *24-43*
DBA_DV_USER_PRIVS view, *24-44*
DBA_DV_USER_PRIVS_ALL view, *24-45*
DBA_USERS_WITH_DEFPWD data dictionary view
access to in Oracle Database Vault, *2-3*
DBMS_FILE_TRANSFER package, guidelines on managing, *D-11*
DBMS_MACADM package
about, *23-1*
command rule procedures, listed, *16-1*
factor procedures, listed, *17-1*
Oracle Label Security policy procedures, listed, *19-1*
realm procedures, listed, *14-1*
rule set procedures, listed, *15-1*
secure application role procedures, listed, *18-1*
DBMS_MACADM PL/SQL package contents, *23-1*
DBMS_MACADM.ADD_APP_EXCEPTION procedure, *21-4*
DBMS_MACADM.ADD_AUTH_TO_REALM procedure, *14-1*
DBMS_MACADM.ADD_CMD_RULE_TO_POLICY procedure, *22-2*, *22-6*
DBMS_MACADM.ADD_FACTOR_LINK procedure, *17-3*
DBMS_MACADM.ADD_NLS_DATA procedure, *C-1*
DBMS_MACADM.ADD_NLS_DATA procedure, *21-5*
DBMS_MACADM.ADD_OBJECT_TO_REALM procedure, *14-4*
DBMS_MACADM.ADD_OWNER_TO_POLICY procedure, *22-3*

DBMS_MACADM.ADD_POLICY_FACTOR procedure, *17-4*
DBMS_MACADM.ADD_REALM_TO_POLICY procedure, *22-4*
DBMS_MACADM.ADD_RULE_TO_RULE_SET procedure, *15-2*
DBMS_MACADM.ALLOW_COMMON_OPERATION procedure, *21-6*
DBMS_MACADM.AUTH_DATAPUMP_CREATE_USER procedure, *21-7*
DBMS_MACADM.AUTH_DATAPUMP_GRANT procedure, *21-8*
DBMS_MACADM.AUTH_DATAPUMP_GRANT_ROLE procedure, *21-8*
DBMS_MACADM.AUTH_DATAPUMP_GRANT_SYSPRIV procedure, *21-9*
DBMS_MACADM.AUTHORIZE_AUDIT_ADMIN procedure, *21-10*
DBMS_MACADM.AUTHORIZE_AUDIT_VIEWER procedure, *21-10*
DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure, *21-11*, *21-29*
DBMS_MACADM.AUTHORIZE_DBCAPTURE procedure, *21-12*
DBMS_MACADM.AUTHORIZE_DBREPLAY procedure, *21-12*
DBMS_MACADM.AUTHORIZE_DDL procedure, *21-13*
DBMS_MACADM.AUTHORIZE_DIAGNOSTIC_ADMIN procedure, *21-14*
DBMS_MACADM.AUTHORIZE_PREPROCESSOR procedure, *21-15*
DBMS_MACADM.AUTHORIZE_PROXY_USER procedure, *21-16*
DBMS_MACADM.AUTHORIZE_SCHEDULER_USER procedure, *21-17*
DBMS_MACADM.AUTHORIZE_SQL_FIREWALL procedure, *21-18*
DBMS_MACADM.AUTHORIZE_TTS_USER procedure, *21-19*
DBMS_MACADM.CHANGE_IDENTITY_FACTOR procedure, *17-4*
DBMS_MACADM.CHANGE_IDENTITY_VALUE procedure, *17-5*
DBMS_MACADM.CREATE_COMMAND_RULE procedure, *16-2*
DBMS_MACADM.CREATE_CONNECT_COMMAND_RULE procedure, *16-9*
DBMS_MACADM.CREATE_DOMAIN_IDENTITY procedure, *17-6*
DBMS_MACADM.CREATE_FACTOR procedure, *17-7*
DBMS_MACADM.CREATE_FACTOR_TYPE procedure, *17-9*
DBMS_MACADM.CREATE_IDENTITY procedure, *17-9*
DBMS_MACADM.CREATE_IDENTITY_MAP procedure, *17-10*
DBMS_MACADM.CREATE_MAC_POLICY procedure, *19-1*
DBMS_MACADM.CREATE_POLICY procedure, *22-5*
DBMS_MACADM.CREATE_POLICY_LABEL procedure, *19-3*
DBMS_MACADM.CREATE_REALM procedure, *14-5*
DBMS_MACADM.CREATE_ROLE procedure, *18-1*
DBMS_MACADM.CREATE_RULE procedure, *15-3*
DBMS_MACADM.CREATE_RULE_SET procedure, *15-5*
DBMS_MACADM.CREATE_SESSION_EVENT_CMD_RULE procedure, *16-11*
DBMS_MACADM.CREATE_SYSTEM_EVENT_CMD_RULE procedure, *16-13*
DBMS_MACADM.DELETE_APP_EXCEPTION procedure, *21-19*
DBMS_MACADM.DELETE_AUTH_FROM_REALM procedure, *14-7*
DBMS_MACADM.DELETE_COMMAND_RULE procedure, *16-14*
DBMS_MACADM.DELETE_CONNECT_COMMAND_RULE procedure, *16-16*
DBMS_MACADM.DELETE_FACTOR procedure, *17-11*
DBMS_MACADM.DELETE_FACTOR_LINK procedure, *17-12*
DBMS_MACADM.DELETE_FACTOR_TYPE procedure, *17-12*
DBMS_MACADM.DELETE_IDENTITY procedure, *17-13*
DBMS_MACADM.DELETE_IDENTITY_MAP procedure, *17-13*
DBMS_MACADM.DELETE_MAC_POLICY_CASCADE procedure, *19-4*
DBMS_MACADM.DELETE_OBJECT_FROM_REALM procedure, *14-8*
DBMS_MACADM.DELETE_OWNER_FROM_POLICY procedure, *22-8*
DBMS_MACADM.DELETE_POLICY_FACTOR procedure, *19-4*
DBMS_MACADM.DELETE_POLICY_LABEL procedure, *19-5*
DBMS_MACADM.DELETE_REALM procedure, *14-9*
DBMS_MACADM.DELETE_REALM_CASCADE procedure, *14-10*

DBMS_MACADM.DELETE_REALM_FROM_POLICY procedure, *22-9*
DBMS_MACADM.DELETE_ROLE procedure, *18-2*
DBMS_MACADM.DELETE_RULE procedure, *15-8*
DBMS_MACADM.DELETE_RULE_FROM_RULE_SET procedure, *15-8*
DBMS_MACADM.DELETE_RULE_SET procedure, *15-9*
DBMS_MACADM.DELETE_SESSION_EVENT_CMD_RULE procedure, *16-17*
DBMS_MACADM.DELETE_SYSTEM_EVENT_CMD_RULE procedure, *16-18*
DBMS_MACADM.DISABLE_APP_PROTECTION procedure, *21-20*
DBMS_MACADM.DISABLE_DV procedure, *21-21*
DBMS_MACADM.DISABLE_DV_DICTIONARY_ACCTS procedure, *21-21*
DBMS_MACADM.DISABLE_DV_PATCH_ADMIN_AUDIT procedure, *21-22*
DBMS_MACADM.DISABLE_ORADEBUG procedure, *21-22*
DBMS_MACADM.DROP_DOMAIN_IDENTITY procedure, *17-14*
DBMS_MACADM.DROP_POLICY procedure, *22-9*
DBMS_MACADM.ENABLE_DV procedure
  about, *21-23*
  configuring and enabling Database Vault with, *3-3*, *3-6*, *3-8*
DBMS_MACADM.ENABLE_DV_DICTIONARY_ACCTS procedure, *21-24*
DBMS_MACADM.ENABLE_ORADEBUG procedure, *21-25*
DBMS_MACADM.ENSABLE_DV_PATCH_ADMIN_AUDIT procedure, *21-25*
DBMS_MACADM.GET_INSTANCE_INFO function, *17-16*
DBMS_MACADM.GET_SESSION_INFO function, *17-15*
DBMS_MACADM.RENAME_FACTOR procedure, *17-16*
DBMS_MACADM.RENAME_FACTOR_TYPE procedure, *17-17*
DBMS_MACADM.RENAME_POLICY procedure, *22-10*
DBMS_MACADM.RENAME_REALM procedure, *14-10*
DBMS_MACADM.RENAME_ROLE procedure, *18-3*
DBMS_MACADM.RENAME_RULE procedure, *15-9*
DBMS_MACADM.RENAME_RULE_SET procedure, *15-10*
DBMS_MACADM.UNAUTH_DATAPUMP_CREATE_USER procedure, *21-26*
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT procedure, *21-26*
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_ROLE procedure, *21-27*
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_SYSPRIV procedure, *21-28*
DBMS_MACADM.UNAUTHORIZE_AUDIT_ADMIN procedure, *21-28*
DBMS_MACADM.UNAUTHORIZE_AUDIT_VIEWER procedure, *21-29*
DBMS_MACADM.UNAUTHORIZE_DBCAPTURE procedure, *21-30*
DBMS_MACADM.UNAUTHORIZE_DBREPLAY procedure, *21-31*
DBMS_MACADM.UNAUTHORIZE_DDL procedure, *21-31*
DBMS_MACADM.UNAUTHORIZE_DIAGNOSTIC_ADMIN procedure, *21-32*
DBMS_MACADM.UNAUTHORIZE_PREPROCESSOR procedure, *21-34*
DBMS_MACADM.UNAUTHORIZE_PROXY_USER procedure, *21-35*
DBMS_MACADM.UNAUTHORIZE_SCHEDULER_USER procedure, *21-35*
DBMS_MACADM.UNAUTHORIZE_SQL_FIREWALL procedure, *21-36*
DBMS_MACADM.UNAUTHORIZE_TTS_USER procedure, *21-37*
DBMS_MACADM.UPDATE_COMMAND_RULE procedure, *16-19*
DBMS_MACADM.UPDATE_CONNECT_COMMAND_RULE procedure, *16-21*
DBMS_MACADM.UPDATE_FACTOR procedure, *17-17*
DBMS_MACADM.UPDATE_FACTOR_TYPE procedure, *17-20*
DBMS_MACADM.UPDATE_IDENTITY procedure, *17-20*
DBMS_MACADM.UPDATE_MAC_POLICY procedure, *19-6*
DBMS_MACADM.UPDATE_POLICY_DESCRIPTION procedure, *22-10*
DBMS_MACADM.UPDATE_POLICY_STATE procedure, *22-11*
DBMS_MACADM.UPDATE_REALM procedure, *14-11*
DBMS_MACADM.UPDATE_REALM_AUTH procedure, *14-13*
DBMS_MACADM.UPDATE_ROLE procedure, *18-3*

DBMS_MACADM.UPDATE_RULE procedure, *15-11*
DBMS_MACADM.UPDATE_RULE_SET procedure, *15-12*
DBMS_MACADM.UPDATE_SESSION_EVENT_CMD_RULE procedure, *16-23*
DBMS_MACADM.UPDATE_SYSTEM_EVENT_CMD_RULE procedure, *16-24*
DBMS_MACSEC_ROLES package
  about, *18-4*
  functions, listed, *18-4*
DBMS_MACSEC_ROLES.CAN_SET_ROLE function, *18-4*
DBMS_MACSEC_ROLES.SET_ROLE procedure, *18-5*
DBMS_MACUTL package
  about, *20-1*
  constants (fields)
    examples, *20-5*
    listed, *20-1*
  procedures and functions, listed, *20-7*
DBMS_MACUTL PL/SQL package contents, *23-7*
DBMS_MACUTL.CHECK_DVSYS_DML_ALLOWED procedure, *20-8*
DBMS_MACUTL.GET_CODE_VALUE function, *20-9*
DBMS_MACUTL.GET_DAY function, *20-11*
DBMS_MACUTL.GET_HOUR function, *20-11*
DBMS_MACUTL.GET_MINUTE function, *20-10*
DBMS_MACUTL.GET_MONTH function, *20-12*
DBMS_MACUTL.GET_SECOND function, *20-9*
DBMS_MACUTL.GET_YEAR function, *20-13*
DBMS_MACUTL.IS_ALPHA function, *20-13*
DBMS_MACUTL.IS_DIGIT function, *20-14*
DBMS_MACUTL.IS_DVSYS_OWNER function, *20-15*
DBMS_MACUTL.IS_OLS_INSTALLED function, *20-15*
DBMS_MACUTL.IS_OLS_INSTALLED_VARCHAR function, *20-16*
DBMS_MACUTL.ROLE_GRANTED_ENABLED_VARCHAR function, *20-16*
DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE function, *20-17*
DBMS_MACUTL.USER_HAS_ROLE function, *20-19*
DBMS_MACUTL.USER_HAS_ROLE_VARCHAR function, *20-19*
DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE function, *20-20*
DBSNMP schema realm protection, *4-7*
DBSNMP user account
  granted DV_MONITOR role, *13-16*
DDL operations
  DV_PATCH_ADMIN impact, *12-3*
  performing in Oracle Database Vault, *12-2*
  removal of default ('%', '%'), *12-4*
  restrictions, *12-2*
  upgrades impact, *12-3*
deinstallation, *B-1*
DELETE_CATALOG_ROLE role, *26-15*
deleting event command rules, *16-17*
Denial of Service (DoS) attacks
  reports
    System Resource Limits Report, *26-16*
    Tablespace Quotas Report, *26-19*
diagnostic view and table queries
  MACADM procedure for authorization, *21-14*
  MACADM procedure for revoking authorization, *21-32*
Direct and Indirect System Privileges By Database Account Report, *26-9*
Direct Object Privileges Report, *26-8*
direct system privileges, *26-9*
Direct System Privileges By Database Account Report, *26-9*
disabling system features with Disabled default rule set, *5-3*
domains
  defined with factors, *7-1*
  finding database domain with DVF.F$DATABASE_DOMAIN, *17-30*
  finding with DVF.F$DOMAIN, *17-32*
DROP ROLE statement
  monitoring, *25-1*
DROP USER statement
  monitoring, *25-1*
dual key connection, dual key security
  See two-person integrity (TPI)
DV_ACCTMGR role, *E-13*
  about, *13-10*
  backup account, *13-25*
  creating profile to protect user granted this role, *3-12*
  Database Vault disabled, *13-10*
  GRANT and REVOKE operations affected by, *13-10*
  privileges associated with, *13-10*
  realm protection, *4-6*
  system privileges of, *13-5*
DV_ADMIN role
  about, *13-12*
  changing password for user granted DV_ADMIN, *13-12*
  Database Vault disabled, *13-12*, *13-17*
  GRANT and REVOKE operations affected by, *13-12*
  privileges associated with, *13-12*

DV_AUDIT_CLEANUP role
about, 13-13
Database Vault disabled, 13-13, 13-16, 13-21
GRANT and REVOKE operations affected by, 13-13
privileges associated with, 13-13
system privileges of, 13-5
DV_DATAPUMP_NETWORK_LINK role
about, 13-14
Database Vault disabled, 13-14
GRANT and REVOKE operations affected by, 13-14
privileges associated with, 13-14
DV_GOLDENDATE_REDO role
privileges associated with, 13-15
DV_GOLDENDGATE_ADMIN role
Database Vault disabled, 13-15
DV_GOLDENGATE_ADMIN role, 13-15
GRANT and REVOKE operations affected by, 13-15
privileges associated with, 13-15
DV_GOLDENGATE_REDO_ACCESS role, 13-15
Database Vault disabled, 13-15
GRANT and REVOKE operations affected by, 13-15
DV_MONITOR role
about, 13-16
Database Vault disabled, 13-16
GRANT and REVOKE operations affected by, 13-16
privileges associated with, 13-16
system privileges of, 13-5
DV_OWNER role, E-12
about, 13-17
backup account, 13-25
changing password for user granted DV_OWNER, 13-17
creating profile to protect user granted this role, 3-12
Database Vault disabled, 13-17
GRANT and REVOKE operations affected by, 13-17
privileges associated with, 13-17
system privileges of, 13-5
DV_PATCH_ADMIN role, 13-19
Database Vault disabled, 13-19
DDL operations impact, 12-3
GRANT and REVOKE operations affected by, 13-19
privileges associated with, 13-19
SYS user, 12-44
DV_POLICY_OWNER role
about, 13-20
DV_POLICY_OWNER role *(continued)*
GRANT and REVOKE operations affected by, 13-20
privileges associated with, 13-20
system privileges of, 13-5
DV_SECANALYST role
about, 13-21
Database Vault disabled, 13-21
GRANT and REVOKE operations affected by, 13-21
privileges associated with, 13-21
system privileges of, 13-5
DV_XSTREAM_ADMIN role, 13-22
Database Vault disabled, 13-22
GRANT and REVOKE operations affected by, 13-22
privileges associated with, 13-22
DVF account
database accounts, 13-23
DVF PL/SQL interface contents, 23-8
DVF schema, 17-26
about, 13-3
DBA_DV_DICTIONARY_ACCTS view, 24-14
PDBs, 13-3
protecting, 21-21
realm protection, 4-6
DVSYS account, 13-23
DVSYS schema
about, 13-1
CDBs, 1-10
DBA_DV_DICTIONARY_ACCTS view, 24-14
DV_OWNER role, 13-17
DV_POLICY_OWNER role, 13-20
PDBs, 13-1, 13-4
protecting, 21-21
realm protection, 4-6
DVSYS.DBA_DV_COMMON_OPERATION_STATUS view, 24-55
DVSYS.DBA_DV_FACTOR_LINK view, 24-17
DVSYS.DV$CONFIGURATION_AUDIT view, 24-46
DVSYS.DV$ENFORCEMENT_AUDIT view, 24-50
DVSYS.DV$REALM view, 24-53
DVSYS.POLICY_OWNER_POLICY view, 24-57
DVSYS.POLICY_OWNER_REALM view, 24-57
DVSYS.POLICY_OWNER_REALM_AUTH view, 24-59
DVSYS.POLICY_OWNER_REALM_OBJECT view, 24-60
DVSYS.POLICY_OWNER_RULE view, 24-61
DVSYS.POLICY_OWNER_RULE_SET view, 24-62
DVSYS.POLICY_OWNER_RULE_SET_RULE view, 24-64

E

ENABLE_APP_PROTECTION procedure, *21-23*
enabling system features with Enabled default rule set, *5-3*
encrypted information, *26-19*
enterprise identities, Enterprise_Identity default factor, *7-2*
Enterprise Manager
  See Oracle Enterprise Manager
Enterprise User Security, integrating with Oracle Database Vault
  about, *11-14*
  configuring, *11-15*
  configuring Database Vault accounts as Enterpriser User accounts, *11-16*
event handler
  rule sets, *5-4*
example, *6-4*
examples, *7-21*
  DBMS_MACUTL constants, *20-5*
  realms, *4-21*
  separation of duty matrix, *D-3*
  trace files, *E-7*–*E-9*
    See also tutorials
Execute Privileges to Strong SYS Packages Report, *26-10*
EXECUTE_CATALOG_ROLE role, *26-15*
  impact of Oracle Database Vault installation, *2-3*
EXEMPT ACCESS POLICY system privilege, *26-14*
exporting data
  See Oracle Data Pump

F

Factor Audit Report, *26-5*
Factor Configuration Issues Report, *26-4*
factor identities
  modifying, *7-15*
Factor Without Identities Report, *26-4*
factors, *7-1*
  about, *7-1*
  assignment
    disabled rule set, *26-4*
    incomplete rule set, *26-4*
  assignment operation, *26-5*
  audit events, custom, *A-3*
  child factors
    Factor Configuration Issues Report, *26-4*
    mapping, *7-13*
  creating, *7-5*
  data dictionary views, *7-29*
  DBA_DV_FACTOR view, *24-14*
factors *(continued)*
  DBA_DV_SIMULATION_LOG view, *24-39*
  DBMS_MACUTL constants, example of, *20-6*
  default factors, *7-2*
  deleting, *7-17*
  domain, finding with DVF.F$DOMAIN, *17-32*
  evaluation operation, *26-5*
  factor-identity pair mapping, *7-13*
  functionality, *7-18*
  functions
    DBMS_MACUTL (utility), *20-1*
    DBMS_MACUTL constants (fields), *20-1*
  guidelines, *7-28*
  identifying using child factors, *7-13*
  identities
    about, *7-9*
    adding to factor, *7-8*
    configuring, *7-12*
    creating, *7-12*
    data dictionary views, *7-29*
    database session, *7-9*
    deleting, *7-16*
    enterprise-wide users, *17-32*
    how factor identities work, *7-9*
    mapping, about, *7-13*
    mapping, procedure, *7-13*
    reports, *7-29*
    setting dynamically, *17-22*
    trust levels, *7-9*, *7-12*
    with Oracle Label Security, *7-9*
  identity maps, deleting, *7-15*
  initialization, command rules, *6-2*
  invalid audit options, *26-4*
  label, *26-4*
  modifying, *7-16*
  performance effect, *7-29*
  procedures
    DBMS_MACADM (configuration), *17-1*
  process flow, *7-18*
  reports, *7-29*
  retrieving, *7-21*
  retrieving with GET_FACTOR, *17-23*
  setting, *7-21*
  setting with SET_FACTOR, *17-22*
  troubleshooting
    auditing report, *26-5*
    configuration problems, *E-11*
    tips, *E-11*
  values (identities), *7-1*
  views
    DBA_DV_FACTOR_LINK, *24-17*
    DBA_DV_FACTOR_TYPE, *24-17*
    DBA_DV_IDENTITY, *24-18*
    DBA_DV_IDENTITY_MAP, *24-19*

factors *(continued)*
views *(continued)*
DBA_DV_MAC_POLICY_FACTOR, *24-21*
ways to assign, *7-9*
FLASHBACK TABLE SQL statement, *4-2*
functions
command rules
DBMS_MACUTL (utility), *20-1*
DVSYS schema enabling, *17-21*
factors
DBMS_MACUTL (utility), *20-1*
Oracle Label Security policy
DBMS_MACADM (configuration), *19-1*
realms
DBMS_MACUTL (utility), *20-1*
rule sets
DBMS_MACADM (configuration), *15-1*
DBMS_MACUTL (utility), *20-1*
PL/SQL functions for inspecting SQL, *15-14*
secure application roles
DBMS_MACADM (configuration), *18-1*
DBMS_MACSEC_ROLES (configuration), *18-4*
DBMS_MACUTL (utility), *20-1*

G

general security reports, *26-6*
GRANT statement
monitoring, *25-1*
guidelines
ALTER SESSION privilege, *D-14*
ALTER SYSTEM privilege, *D-14*
backup DV_OWNER and DV_ACCTMGR accounts, *13-25*
command rules, *6-15*
CREATE ANY JOB privilege, *D-14*
CREATE EXTERNAL JOB privilege, *D-14*
CREATE JOB privilege, *D-14*
DBMS_FILE_TRANSFER package, *D-11*
factors, *7-28*
general security, *D-1*
operating system access, *D-6*
Oracle software owner, *D-8*
performance effect, *7-29*
realms, *4-22*
root access, *D-6*
root user access, *D-8*
rule sets, *5-21*
secure application roles, *8-2*
SYSDBA access, *D-9*
SYSDBA privilege, limiting, *D-6*
SYSOPER access, *D-9*
guidelines *(continued)*
SYSTEM schema and application tables, *D-6*
SYSTEM user account, *D-5*
trusted accounts and roles, *D-7*
using Database Vault in a production environment, *D-10*
UTL_FILE package, *D-11*

H

hackers
See security attacks
Hierarchical System Privileges by Database Account Report, *26-9*
host names
finding with
DVF.F$DATABASE_HOSTNAME, *17-30*

I

identities
See factors, identities
Identity Configuration Issues Report, *26-4*
IDLE_TIME resource profile, *26-16*
IMP_FULL_DATABASE role
impact of Oracle Database Vault installation, *2-3*
importing data
See Oracle Data Pump
incomplete rule set, *26-4*
role enablement, *26-4*
Information Lifecycle Management, *4-2*
authorizations, about, *12-22*
granting users authorization for, *12-22*
revoking authorization from users, *12-23*
initialization parameters
Allow System Parameters default rule set, *5-3*
modified after installation, *2-1*
modified by Oracle Database Vault, *2-1*
reports, *26-16*
insider threats
See intruders
installations
Database Vault and Label Security in a multitenant environment, *3-13*
security considerations, *D-10*
intruders, *26-12*, *26-18*
compromising privileged accounts, *1-8*
See also security attacks
IP addresses
Client_IP default factor, *7-2*
defined with factors, *7-1*

J

Java Policy Grants Report, *26-18*
jobs, scheduling
  See Oracle Scheduler

L

Label Security Integration Audit Report, *26-6*
labels, *7-11*
  about, *7-11*
    See also Oracle Label Security
languages
  adding to Oracle Database Vault, *C-1*
  finding with DVF.F$LANG, *17-35*
  finding with DVF.F$LANGUAGE, *17-35*
  name
    Lang default factor, *7-2*
    Language default factor, *7-2*
LBACSYS account, *13-23*
  about, *13-23*
    See also Oracle Label Security
LBACSYS schema
  realm protection, *4-6*
locked out accounts, solution for, *B-1*
log files
  Database Vault log files, *A-4*
logging on
  reports, Core Database Audit Report, *26-17*

M

managing user accounts and profiles
  Can Maintain Accounts/Profiles default rule set, *5-3*
managing user accounts and profiles on own account, Can Maintain Own Accounts default rule set, *5-3*
mandatory realms
  about, *4-3*
mapping identities, *7-13*
MDDATA schema realm protection, *4-7*
MDSYS schema realm protection, *4-7*
modules
  function to return information about, *17-33*
monitoring
  activities, *25-1*
multitenant container databases
  See CDBs

N

naming conventions
  realms, *4-10*
  rule sets, *5-4*
naming conventions *(continued)*
  rules, *5-9*
network protocol
  finding with DVF.F$NETWORK_PROTOCOL, *17-36*
network protocol, Network_Protocol default factor, *7-2*
NOAUDIT statement
  monitoring, *25-1*
Non-Owner Object Trigger Report, *26-19*
nonsystem database accounts, *26-8*

O

Object Access By PUBLIC Report, *26-7*
Object Access Not By PUBLIC Report, *26-8*
Object Dependencies Report, *26-8*
object owners
  nonexistent, *26-3*
  reports
    Command Rule Configuration Issues Report, *26-3*
object privilege reports, *26-7*
object types
  supported for Database Vault realm protection, *4-5*
objects, *13-1*, *24-32*
  command rule objects
    name, *6-9*
    owner, *6-9*
    processing, *6-12*
  dynamic SQL use, *26-18*
  mandatory realms, *4-3*
  monitoring, *25-1*
  object names
    finding with DV_DICT_OBJ_NAME, *15-17*
  object owners
    finding with DV_DICT_OBJ_OWNER, *15-17*
  realms
    object name, *4-10*
    object owner, *4-10*
    object type, *4-10*
    procedures for registering, *14-4*
  reports
    Access to Sensitive Objects Report, *26-11*
    Accounts with SYSDBA/SYSOPER Privilege Report, *26-12*
    Direct Object Privileges Report, *26-8*
    Execute Privileges to Strong SYS Packages Report, *26-10*
    Non-Owner Object Trigger Report, *26-19*

objects *(continued)*
reports *(continued)*
Object Access By PUBLIC Report, *26-7*
Object Access Not By PUBLIC Report, *26-8*
Object Dependencies Report, *26-8*
Objects Dependent on Dynamic SQL Report, *26-18*
OS Directory Objects Report, *26-18*
privilege, *26-7*
Public Execute Privilege To SYS PL/SQL Procedures Report, *26-11*
sensitive, *26-10*
System Privileges By Privilege Report, *26-10*
restricting user access to using mandatory realms, *4-3*
types
finding with DV_DICT_OBJ_TYPE, *15-16*
views, DBA_DV_REALM_OBJECT, *24-32*
See also database objects
Objects Dependent on Dynamic SQL Report, *26-18*
OEM
See Oracle Enterprise Manager (OEM)
OEM_MONITOR schema realm protection, *4-7*
OLS
See Oracle Label Security
operating system access
guideline for using with Database Vault, *D-6*
operating systems
reports
OS Directory Objects Report, *26-18*
OS Security Vulnerability Privileges Report, *26-16*
vulnerabilities, *26-16*
ORA_DV_DEFAULT_PROTECTION predefined unified audit policy, *A-1*, *A-8*
ORA_DV_SCHEMA_CHANGES predefined unified audit policy, *A-1*, *A-8*
ORA-00942 error, *8-9*
ORA-01301 error, *12-35*
ORA-06512 error, *20-8*
ORA-47305 error, *8-9*
ORA-47400 error, *12-35*
ORA-47401 error, *4-20*, *12-35*
ORA-47408 error, *12-35*
ORA-47409 error, *12-35*
ORA-47500 error, *21-38*
ORA-47503 error, *3-6*, *3-8*
ORA-47920 error, *20-8*
Oracle APEX, integrating with Oracle Database Vault, *11-17*
about, *11-18*
Oracle APEX, integrating with Oracle Database Vault *(continued)*
authorizing APEX schema, *11-19*
authorizing Oracle Scheduler, *11-20*
DDL tasks, *11-20*
installing or upgrading Oracle APEX, *11-18*
maintenance tasks, *11-21*
protected objects, *11-22*
proxy users for ORDS, *11-21*
troubleshooting, *11-22*
Oracle Data Guard
integrating Database Vault with, *11-11*
Oracle Data Guard, integrating with Oracle Database Vault
configuring primary database, *11-12*
configuring standby database, *11-12*
disabling Oracle Database Vault, *11-13*
how auditing works after integration, *11-13*
Oracle Data Pump, *12-10*
archiving the Oracle Database Vault audit trail with, *A-6*
authorizing transportable tablespace operations for Database Vault, *12-16*
DBA_DV_DATAPUMP_AUTH view, *24-11*
DBA_DV_TTS_AUTH view, *24-43*
DBMS_MACADM.AUTH_DATAPUMP_CREATE_USER procedure, *21-7*
DBMS_MACADM.AUTH_DATAPUMP_GRANT procedure, *21-8*
DBMS_MACADM.AUTH_DATAPUMP_GRANT_ROLE procedure, *21-8*
DBMS_MACADM.AUTH_DATAPUMP_GRANT_SYSPRIV procedure, *21-9*
DBMS_MACADM.AUTHORIZE_TTS_USER, *21-19*
DBMS_MACADM.UNAUTH_DATAPUMP_CREATE_USER procedure, *21-26*
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT procedure, *21-26*
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_ROLE procedure, *21-27*
DBMS_MACADM.UNAUTH_DATAPUMP_GRANT_SYSPRI procedure, *21-28*
DBMS_MACADM.UNAUTHORIZE_TTS_USER, *21-37*
granting authorization to use with Database Vault, *12-12*
guidelines before performing an export or import, *12-18*
levels of authorization required
Oracle Data Pump only, *12-11*
transportable tablespaces, *12-15*
MACADM procedure for authorization, *21-11*
realm protection, *4-8*
revoking standard authorization, *12-13*
revoking transportable tablespace authorization, *12-17*
using with Oracle Database Vault, *12-10*
Oracle Database Replay
authorizations, about, *12-24*
Database Vault authorization
granting for workload captures, *21-12*

Oracle Database Replay *(continued)*
Database Vault authorization *(continued)*
granting for workload replays, *21-12*
revoking for workload captures, *21-30*
revoking for workload replays, *21-31*
granting users authorization for workload capture operations, *12-24*
granting users authorization for workload replay operations, *12-24*
revoking workload capture authorization from users, *12-25*
revoking workload replay authorization from users, *12-26*
Oracle Database Vault, *1-2*
about, *1-2*
components, *1-4*, *1-5*
configuring and enabling
using DBCA, *3-1*
disabling
procedures for, *B-1*
reasons for, *B-1*
enabling
procedures for, *B-1*
integrating with other Oracle products, *11-1*
Oracle Database installation, affect on, *2-1*
post-installation procedures, *C-1*
privileges to use, *1-4*
reinstalling, *C-3*
roles
system privileges of, *13-5*
uninstalling, *C-2*
Oracle Database Vault accounts
created during registration, *13-23*
Oracle Database Vault Administrator (DVA)
logging on from Oracle Enterprise Manager Cloud Control, *3-15*
Oracle Database Vault Administrator pages, *1-7*
Oracle Database Vault configuration and enablement
common user to manage CDB root, *3-3*
common users to manage specific PDBs, *3-6*
local users to manage specific PDBs, *3-8*
Oracle Database Vault configuring and enabling
about, *3-1*
Oracle Database Vault operations control
about, *12-28*
Oracle Database Vault policies, *9-1*
about, *9-1*
creating, *9-3*
data dictionary views, *9-7*
default, *9-3*
deleting, *9-6*
in multitenant environment, *9-3*
modifying, *9-6*
Oracle Database Vault realm, *4-6*
Oracle Database Vault registration
creating profile to protect DV_OWNER and DV_ACCTMGR users, *3-12*
verifying configuration and enablement, *3-14*
Oracle Default Component Protection Realm, *4-9*
Oracle Default Schema Protection Realm, *4-7*
Oracle Enterprise Manager, *4-7*
DBSNMP account
granted DV_MONITOR role, *13-16*
using Oracle Database Vault with, *12-7*
Oracle Enterprise Manager Cloud Control
monitoring Database Vault for attempted violations, *13-16*
propagating Database Vault configurations to other databases, *12-7*
starting Oracle Database Vault from, *3-15*
Oracle Enterprise Manager realm, *4-7*
Oracle Enterprise User Security, integrating with Oracle Database Vault, *11-14*
Oracle Flashback Technology, *4-2*, *6-2*
Oracle GoldenGate
Database Vault role used for
DV_GOLDENGATE_ADMIN, *13-15*
DV_GOLDENGATE_REDO_ACCESS, *13-15*
in an Oracle Database Vault environment, *12-33*
Oracle GoldenGate Protection Realm, *4-9*
Oracle Internet Directory Distinguished Name, Proxy_Enterprise_Identity default factor, *7-2*
Oracle Internet Directory, integrating with Oracle Database Vault, *11-14*
Oracle Label Security, *7-11*, *13-23*
using OLS_LABEL_DOMINATES function in rule expressions, *15-3*
Oracle Label Security (OLS), *13-23*
audit events, custom, *A-3*
checking if installed using DBMS_MACUTL functions, *20-7*
functions
DBMS_MACUTL (utility), *20-1*
initialization, command rules, *6-2*
integration with Oracle Database Vault
example, *11-7*
Label Security Integration Audit Report, *26-6*
labels
about, *7-11*
determining with GET_FACTOR_LABEL, *17-23*
invalid label identities, *26-4*
policies
accounts that bypass, *26-14*
monitoring policy changes, *25-1*

Oracle Label Security (OLS) *(continued)*
policies *(continued)*
nonexistent, *26-4*
procedures
DBMS_MACADM (configuration), *19-1*
views
DBA_DV_MAC_POLICY, *24-20*
DBA_DV_MAC_POLICY_FACTOR, *24-21*
DBA_DV_POLICY_LABEL, *24-24*
See also LBACSYS account
Oracle Label Security realm, *4-9*
Oracle Label Security, integrating with Oracle Database Vault
about, *11-3*
data dictionary views, *11-10*
factors, about, *11-4*
factors, configuring, *11-5*
reports, *11-10*
requirements, *11-3*
tutorial, *11-7*
Oracle Real Application Clusters
configuring and enabling Database Vault on Oracle RAC nodes, *3-11*
multiple factor identities, *7-9*
uninstalling Oracle Database Vault from, *C-2*
Oracle Recovery Manager (RMAN)
in an Oracle Database Vault environment, *12-32*
Oracle Scheduler, *12-19*
DBA_DV_JOB_AUTH view, *24-20*
granting Oracle Database Vault authorization, *12-20*
realm protection, *4-8*
revoking Oracle Database Vault authorization, *12-21*
SCHEDULER_ADMIN role, impact of Oracle Database Vault installation, *2-3*
using with Oracle Database Vault, *12-19*
Oracle software owner, guidelines on managing, *D-8*
Oracle Spatial realm protection, *4-7*
Oracle System Privilege and Role Management Realm, *4-8*
Oracle Text realm protection, *4-7*
Oracle Virtual Private Database (VPD), *5-3*
accounts that bypass, *26-14*
GRANT EXECUTE privileges with Grant VPD Administration default rule set, *5-3*
using Database Vault factors with Oracle Label Security, *11-7*
Oracle Virtual Private Database, integrating with Oracle Database Vault
attaching factors to VPD, *11-2*
ORADEBUG utility
about, *12-44*
DBA_DV_ORADEBUG view, *24-22*
PL/SQL procedure for disabling in Database Vault, *21-22*
PL/SQL procedure for enabling in Database Vault, *21-25*
using with Database Vault, *12-44*
OS Directory Objects Report, *26-18*
OS Security Vulnerability Privileges Report, *26-16*
OS_ROLES initialization parameter, *2-1*
OUTLN schema realm protection, *4-9*

P

parameters
modified after installation, *2-1*
reports
Security Related Database Parameters Report, *26-16*
Password History Access Report, *26-15*
passwords
forgotten, solution for, *B-1*
reports, *26-17*
Database Account Default Password Report, *26-17*
Password History Access Report, *26-15*
Username/Password Tables Report, *26-19*
resetting for DV_ACCTMGR user, *E-13*
resetting for DV_OWNER user, *E-12*
patch operations in Database Vault environment, *12-44*
patches
auditing DV_PATCH_ADMIN user, *13-19*
DBMS_MACADM.DISABLE_DV_PATCH_ADMIN_AUDIT procedure, *21-22*
DBMS_MACADM.ENSABLE_DV_PATCH_ADMIN_AUDIT procedure, *21-25*
DV_PATCH_ADMIN requirement for, *13-19*
security consideration, *D-10*
two-person integrity used for, *5-16*
PDBs, *1-10*
command rules in, *6-3*
disabling tracing
all database sessions, *E-11*
current database session, *E-11*
DVF schema, *13-3*
DVSYS schema, *13-1*, *13-4*
enabling tracing
all database sessions, *E-5*
current database session, *E-3*
plugging Database Vault-enabled PDB to CDB, *12-41*

performance effect
command rules, 6-16
realms, 4-24
reports
Resource Profiles Report, 26-16
System Resource Limits Report, 26-16
rule sets, 5-22
secure application roles, 8-10
static evaluation for rule sets, 5-22
performance tools
Automatic Workload Repository (AWR)
command rules, 6-16
factors, 7-29
Oracle Enterprise Manager performance tools, 4-24
performance tools
Cloud Control, realms, 4-24
Oracle Enterprise Manager realms, 4-24
realms, 4-24
rule sets, 5-22
secure application roles, 8-10
Oracle Enterprise Manager
command rules, 6-16
factors, 7-29
performance tools
Oracle Enterprise Manager Cloud Control command rules, 6-16
rule sets, 5-22
secure application roles, 8-10
Oracle Enterprise Manager Cloud Control
factors, 7-29
rule sets, 5-22
secure application roles, 8-10
TKPROF utility
command rules, 6-16
factors, 7-29
realms, 4-24
rule sets, 5-22
secure application roles, 8-10
PL/SQL
packages
unwrapped bodies, 26-19
Unwrapped PL/SQL Package Bodies Report, 26-19
PL/SQL factor functions, 17-26
pluggable databases
See PDBs
policies
See Oracle Database Vault policies
policy changes, monitoring, 25-1
POLICY_OWNER_COMMAND_RULE view, 24-55
post-installation procedures, C-1
predefined unified audit policies
ORA_DV_DEFAULT_PROTECTION, A-8
ORA_DV_SCHEMA_CHANGES, A-8
preprocessor programs
about executing in Database Vault environment, 12-26
authorizing users in Database Vault environment, 12-27
Database Vault authorization
granting, 21-15
revoking, 21-34
revoking authorization from Database Vault users, 12-27
privileges
checking with DBMS_MACUTL.USER_HAS_OBJECT_PRIVILEGE function, 20-7
existing users and roles, Database Vault affect on, 2-3
least privilege principle
violations to, 26-18
monitoring
GRANT statement, 25-1
REVOKE statement, 25-1
Oracle Database Vault restricting, 2-2
prevented from existing users and roles, 2-5
reports
Accounts With DBA Roles Report, 26-14
ALTER SYSTEM or ALTER SESSION Report, 26-14
ANY System Privileges for Database Accounts Report, 26-10
AUDIT Privileges Report, 26-16
Database Accounts With Catalog Roles Report, 26-15
Direct and Indirect System Privileges By Database Account Report, 26-9
Direct System Privileges By Database Account Report, 26-9
Hierarchical System Privileges By Database Account Report, 26-9
listed, 26-12
OS Directory Objects Report, 26-18
Privileges Distribution By Grantee Report, 26-12
Privileges Distribution By Grantee, Owner Report, 26-12
Privileges Distribution By Grantee, Owner, Privilege Report, 26-13
WITH GRANT Privileges Report, 26-15
restricting access using mandatory realms, 4-3
roles
checking with DBMS_MACUTL.USER_HAS_ROLE_VARCHAR function, 20-7

privileges *(continued)*
system
checking with DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEGE function, *20-7*
views
DBA_DV_PUB_PRIVS, *24-28*
DBA_DV_USER_PRIVS, *24-44*
DBA_DV_USER_PRIVS_ALL, *24-45*
Privileges Distribution By Grantee Report, *26-12*
Privileges Distribution By Grantee, Owner Report, *26-12*
Privileges Distribution By Grantee, Owner, Privilege Report, *26-13*
privileges using external password, *26-12*
problems, diagnosing, *E-2*
procedures
command rules
.DBMS_MACADM (configuration), *16-1*
factors
DBMS_MACADM (configuration), *17-1*
realms
DBMS_MACADM (configuration), *14-1*
production environments
guidelines for securing, *D-10*
profiles, *26-16*
proxy user authorization
Database Vault authorization
DBA_DV_PROXY_AUTH view, *24-27*
granting, *21-16*
revoking, *21-35*
proxy users
function to return name of, *17-37*
PUBLIC access to realms, *4-17*
Public Execute Privilege To SYS PL/SQL Procedures Report, *26-11*
PUBLIC user account
impact of Oracle Database Vault installation, *2-3*

Q

quotas
tablespace, *26-19*

R

Realm Audit Report, *26-5*
Realm Authorization Configuration Issues Report, *26-3*
realm authorizations:multitenant environment, *4-16*
realms, *4-10*
about, *4-2*
adding roles to as grantees, *4-22*
realms *(continued)*
audit events, custom, *A-3*
authentication-related procedures, *14-1*
authorization
enabling access to realm-protected objects, *4-21*
how realm authorizations work, *4-19*
process flow, *4-19*
troubleshooting, *E-11*
authorizations
grantee, *4-10*
rule set, *4-10*
authorizations in multitenant environment, *4-16*
creating, *4-10*
creating names, *4-10*
data dictionary views, *4-24*
data masking, *12-36*
Database Vault Account Management realm, *4-6*
DBMS_MACUTL constants, example of, *20-5*
default realms
listed, *4-5*
deleting, *4-14*
effect on other Oracle Database Vault components, *4-22*
enabling access to realm-protected objects, *4-21*
example, *4-21*
functions
DBMS_MACUTL (utility), *20-1*
DBMS_MACUTL constants (fields), *20-1*
guidelines, *4-22*
how realms work, *4-17*
mandatory realms, *4-3*
modifying, *4-13*
multitenant environment
about, *4-4*
naming conventions, *4-10*
object types, supported, *4-5*
object-related procedures, *14-4*
Oracle Audit realm, *4-10*
Oracle Database Vault realm, *4-6*
Oracle Default Component Protection Realm, *4-9*
Oracle Default Schema Protection Realm, *4-7*
Oracle Enterprise Manager realm, *4-7*
Oracle GoldenGate Protection Realm, *4-9*
Oracle Label Security realm, *4-9*
Oracle System Privilege and Role Management Realm, *4-8*
performance effect, *4-24*
procedures
DBMS_MACADM (configuration), *14-1*

realms *(continued)*
process flow, *4-17*
propagating configuration to other databases, *12-7*
protection after object is dropped, *4-22*
PUBLIC access, *4-17*
realm authorizations
about, *4-16*
realm secured objects
object name, *4-10*
object owner, *4-10*
object type, *4-10*
realm-secured objects, *4-15*
reports, *4-24*
secured object, *26-3*
simulation mode, *10-1*
territory a realm protects, *4-15*
troubleshooting, *E-11*
tutorial, *3-17*
views
DBA_DV_CODE, *24-8*
DBA_DV_MAINTENANCE_AUTH, *24-21*
DBA_DV_POLICY, *24-23*
DBA_DV_POLICY_OBJECT, *24-25*
DBA_DV_POLICY_OWNER, *24-26*
DBA_DV_REALM, *24-29*
DBA_DV_REALM_OBJECT, *24-32*
DBS_DV_REALM_AUTH, *24-31*
DVSYS.POLICY_OWNER_COMMAND_RULE, *24-55*
DVSYS.POLICY_OWNER_POLICY, *24-57*
DVSYS.POLICY_OWNER_REALM, *24-57*
DVSYS.POLICY_OWNER_REALM_AUTH, *24-59*
DVSYS.POLICY_OWNER_REALM_OBJECT, *24-60*
DVSYS.POLICY_OWNER_RULE, *24-61*
DVSYS.POLICY_OWNER_RULE_SET, *24-62*
DVSYS.POLICY_OWNER_RULE_SET_RULE, *24-64*
See also rule sets
recovering lost password, *E-12*, *E-13*
RECOVERY_CATALOG_OWNER role, *26-15*
RECYCLEBIN initialization parameter
default setting in Oracle Database Vault, *2-1*
reinstalling Oracle Database Vault, *C-3*
REMOTE_LOGIN_PASSWORDFILE initialization parameter, *2-1*
reports
about, *26-1*
Access to Sensitive Objects Report, *26-11*
Accounts With DBA Roles Report, *26-14*
Accounts with SYSDBA/SYSOPER Privilege Report, *26-12*
reports *(continued)*
ALTER SYSTEM or ALTER SESSION Report, *26-14*
ANY System Privileges for Database Accounts Report, *26-10*
AUDIT Privileges Report, *26-16*
auditing, *26-5*
BECOME USER Report, *26-14*
categories of, *26-1*
Command Rule Audit Report, *26-5*
Command Rule Configuration Issues Report, *26-3*
Core Database Audit Report, *26-17*
Core Database Vault Audit Trail Report, *26-6*
Database Account Default Password Report, *26-17*
Database Account Status Report, *26-17*
Database Accounts With Catalog Roles Report, *26-15*
Direct and Indirect System Privileges By Database Account Report, *26-9*
Direct Object Privileges Report, *26-8*
Direct System Privileges By Database Account Report, *26-9*
Enterprise Manager Cloud Control, *12-9*
Execute Privileges to Strong SYS Packages Report, *26-10*
Factor Audit Report, *26-5*
Factor Configuration Issues Report, *26-4*
Factor Without Identities, *26-4*
general security, *26-6*
Hierarchical System Privileges by Database Account Report, *26-9*
Identity Configuration Issues Report, *26-4*
Java Policy Grants Report, *26-18*
Label Security Integration Audit Report, *26-6*
Non-Owner Object Trigger Report, *26-19*
Object Access By PUBLIC Report, *26-7*
Object Access Not By PUBLIC Report, *26-8*
Object Dependencies Report, *26-8*
Objects Dependent on Dynamic SQL Report, *26-18*
OS Directory Objects Report, *26-18*
OS Security Vulnerability Privileges, *26-16*
Password History Access Report, *26-15*
permissions for running, *26-2*
privilege management, *26-12*
Privileges Distribution By Grantee Report, *26-12*
Privileges Distribution By Grantee, Owner Report, *26-12*
Privileges Distribution By Grantee, Owner, Privilege Report, *26-13*
Public Execute Privilege To SYS PL/SQL Procedures Report, *26-11*

reports *(continued)*
Realm Audit Report, *26-5*
Realm Authorization Configuration Issues Report, *26-3*
Resource Profiles Report, *26-16*
Roles/Accounts That Have a Given Role Report, *26-15*
Rule Set Configuration Issues Report, *26-3*
running, *26-2*
Secure Application Configuration Issues Report, *26-4*
Secure Application Role Audit Report, *26-6*
Security Policy Exemption Report, *26-14*
Security Related Database Parameters, *26-16*
security vulnerability, *26-17*
System Privileges By Privilege Report, *26-10*
System Resource Limits Report, *26-16*
Tablespace Quotas Report, *26-19*
Unwrapped PL/SQL Package Bodies Report, *26-19*
Username /Password Tables Report, *26-19*
WITH ADMIN Privileges Grants Report, *26-14*
WITH GRANT Privileges Report, *26-15*
Resource Profiles Report, *26-16*
resources
reports
Resource Profiles Report, *26-16*
System Resource Limits Report, *26-16*
REVOKE statement
monitoring, *25-1*
roles, *8-1*
adding to realms as grantees, *4-22*
catalog-based, *26-15*
Database Vault default roles, *13-4*
privileges, checking with DBMS_MACUTL.USER_HAS_ROLE_VARCHAR function, *20-7*
role enablement in incomplete rule set, *26-4*
role-based system privileges, *26-9*
See also secure application roles
Roles/Accounts That Have a Given Role Report, *26-15*
root access
guideline for using with Database Vault, *D-6*
guidelines on managing, *D-8*
Rule Set Configuration Issues Report, *26-3*
rule sets, *4-10*, *5-1*, *5-7*, *6-2*, *6-9*
about, *5-1*
adding existing rules, *5-11*
audit options, *5-4*
auditing
intruders
using rule sets, *5-4*
rule sets *(continued)*
command rules
disabled, *26-3*
selecting for, *6-9*
used with, *6-2*
creating, *5-4*
rules in, *5-9*
creating names, *5-4*
data dictionary views, *5-22*
DBMS_MACUTL constants, example of, *20-6*
default rule sets, *5-3*
default rules, *5-8*
default, no longer supported, *5-2*
deleting, *5-14*
rules from, *5-12*
disabled for
factor assignment, *26-4*
realm authorization, *26-3*
evaluation of rules, *5-7*
event handlers, *5-4*
events firing, finding with DV_SYSEVENT, *15-15*
fail code, *5-4*
fail message, *5-4*
functions
DBMS_MACADM (configuration), *15-1*
DBMS_MACUTL (utility), *20-1*
DBMS_MACUTL constants (fields), *20-1*
PL/SQL functions for rule sets, *15-14*
guidelines, *5-21*
how rule sets work, *5-15*
incomplete, *26-3*
modifying, *5-13*
multitenant environment
about, *5-2*
naming conventions, *5-4*
nested rules, *5-15*
performance effect, *5-22*
procedures
DBMS_MACADM (configuration), *15-1*
process flow, *5-15*
propagating configuration to other databases, *12-7*
reports, *5-22*
rule sets, *4-10*, *5-1*, *5-7*, *6-2*, *6-9*
evaluation options, *5-4*
rules that exclude one user, *5-16*
security attacks, *26-18*
tracking
with rule set auditing, *5-4*
static evaluation, *5-21*
troubleshooting, *E-11*
views
DBA_DV_RULE, *24-34*
DBA_DV_RULE_SET, *24-35*

rule sets *(continued)*
views *(continued)*
DBA_DV_RULE_SET_RULE, *24-38*
See also command rules, factors, realms, rules, secure application roles
rules, *5-7*
about, *5-7*
creating, *5-9*
creating names, *5-9*
data dictionary views, *5-22*
default, *5-8*
default, no longer supported, *5-2*
deleting, *5-12*
deleting from rule set, *5-12*
existing rules, adding to rule set, *5-11*
modifying, *5-12*
naming conventions, *5-9*
nested within a rule set, *5-15*
removing from rule set, *5-12*
reports, *5-22*
troubleshooting, *E-11*
views
DBA_DV_RULE, *24-34*
DBA_DV_RULE_SET_RULE, *24-38*
See also rule sets
rules sets
audit event, custom, *A-3*

S

SCHEDULER_ADMIN role
impact of Oracle Database Vault installation, *2-3*
scheduling database jobs
CREATE EXTERNAL JOB privilege security consideration, *D-14*
scheduling jobs
See Oracle Scheduler
schemas
DVF, *13-3*
DVSYS, *13-1*
Secure Application Configuration Issues Report, *26-4*
secure application role, *8-1*
Secure Application Role Audit Report, *26-6*
secure application roles, *8-1*
audit event, custom, *A-3*
creating, *8-2*
data dictionary view, *8-10*
DBMS_MACSEC_ROLES.SET_ROLE function, *8-2*
deleting, *8-5*
enabling Oracle Database roles to work with Oracle Database Vault, *8-4*
functionality, *8-5*
secure application roles *(continued)*
functions
DBMS_MACADM (configuration), *18-1*
DBMS_MACSEC_ROLES (configuration), *18-4*
DBMS_MACSEC_ROLES package, *18-4*
DBMS_MACUTL (utility), *20-1*
DBMS_MACUTL constants (fields), *20-1*
guidelines on managing, *8-2*
modifying, *8-4*
performance effect, *8-10*
procedure
DBMS_MACADM (configuration), *18-1*
procedures and functions
DBMS_MACUTL (utility), *20-7*
propagating configuration to other databases, *12-7*
reports, *8-10*
Rule Set Configuration Issues Report, *26-3*
troubleshooting, *E-11*
troubleshooting with auditing report, *26-6*
tutorial, *8-6*
views
DBA_DV_ROLE, *24-33*
See also roles, rule sets
security attacks, *26-18*
Denial of Service (DoS) attacks
finding system resource limits, *26-16*
Denial of Service attacks
finding tablespace quotas, *26-19*
eliminating audit trail, *26-16*
monitoring security violations, *25-1*
Oracle Database Vault addressing compromised privileged user accounts, *1-8*
reports
AUDIT Privileges Report, *26-16*
Objects Dependent on Dynamic SQL Report, *26-18*
Privileges Distribution By Grantee, Owner Report, *26-12*
Unwrapped PL/SQL Package Bodies Report, *26-19*
SQL injection attacks, *26-18*
security policies, Oracle Database Vault addressing, *1-8*
Security Policy Exemption Report, *26-14*
Security Related Database Parameters Report, *26-16*
security violations
monitoring attempts, *25-1*
security vulnerabilities
how Database Vault addresses, *1-9*
operating systems, *26-16*

security vulnerabilities *(continued)*
reports, *26-17*
Security Related Database Parameters Report, *26-16*
root operating system directory, *26-18*
SELECT_CATALOG_ROLE role, *26-15*
sensitive objects reports, *26-10*
separation of duty concept
about, *D-1*
command rules, *6-6*
database accounts, suggested, *13-24*
database roles, *2-3*
documenting tasks, *D-4*
example matrix, *D-3*
how Oracle Database Vault addresses, *2-3*
realms, *1-9*
restricting privileges, *2-2*
roles, *13-4*
tasks in Oracle Database Vault environment, *D-2*
session event command rule
updating, *16-23*
session event command rules
creating for events, *16-11*
deleting, *16-17*
sessions
audit events, custom, *A-3*
DBMS_MACUTL fields, *20-1*
finding session user with DVF.F$SESSION_USER, *17-38*
retrieving information with functions, *17-1*
simulation mode
about, *10-1*
use cases, *10-2*
simulation mode, realms
considerations, *10-4*
use cases
adding authorized users to a realm, *10-7*
adding new objects to a realm, *10-7*
all in simulation mode, *10-4*
new realms introduced to existing realms, *10-5*
removing authorized users from a realm, *10-7*
removing objects from a realm, *10-7*
testing new changes to an existing command rule, *10-8*
testing new factors with realms, *10-8*
SQL Firewall
DBMS_MACADM.AUTHORIZE_SQL_FIREWALL procedure, *21-18*
DBMS_MACADM.UNAUTHORIZE_SQL_FIREWALL procedure, *21-36*
granting authorization to use with Database Vault, *12-5*
SQL Firewall *(continued)*
revoking authorization to use with Database Vault, *12-6*
using with Oracle Database Vault, *12-5*
SQL injection attacks, detecting with Object Dependent on Dynamic SQL Report, *26-18*
SQL statements
default command rules that protect, *6-6*
SQL statements protected by, *6-7*
SQL text, finding with DV_SQL_TEXT, *15-18*
SQL92_SECURITY initialization parameter, *2-1*
subfactors
See child factors under factors topic
SYS user account
adding to realm authorization, *4-22*
SYS user, patch operations, *12-44*
SYSDBA access
guidelines on managing, *D-9*
SYSDBA privilege
limiting, importance of, *D-6*
SYSOPER access
guidelines on managing, *D-9*
system event command rule
updating, *16-24*
system event command rules
creating, *16-13*
deleting, *16-18*
system features
disabling with Disabled rule set, *5-3*
enabling with Enabled rule set, *5-3*
system privileges
checking with DBMS_MACUTL.USER_HAS_SYSTEM_PRIVILEG function, *20-7*
Oracle Database Vault roles, *13-5*
reports
System Privileges By Privileges Report, *26-10*
System Privileges By Privilege Report, *26-10*
System Resource Limits Report, *26-16*
system root access, guideline on managing, *D-8*
SYSTEM schema
application tables in, *D-6*
realm protection, *4-9*
SYSTEM user account
guidelines for using with Database Vault, *D-5*

T

tablespace quotas, *26-19*
Tablespace Quotas Report, *26-19*
time data
DBMS_MACUTL functions, *20-7*
trace files
about, *E-2*

trace files, Oracle Database Vault
about, E-2
activities that can be traced, E-2
ADRCI utility, E-6
directory location for trace files, E-6
disabling for all sessions, E-10
disabling for current session, E-10
enabling for all sessions, E-4
enabling for current session, E-3
examples
high level authorization, E-8
highest level on realm violations, E-9
low level realm violations, E-7
finding trace file directory, E-6
levels of trace events, E-3
performance effect, E-3
querying
ADRCI utility, E-6
Linux grep command, E-6
traditional auditing
in Oracle Database Vault, A-2
traisimulationning mode
tutorial, 10-8
Transparent Data Encryption, integrating with Oracle Database Vault, 11-1
transportable tablespaces
authorizing for Oracle Data Pump operations in Database Vault, 12-16
DBA_DV_TTS_AUTH view, 24-43
DBMS_MACADM.AUTHORIZE_TTS_USER procedure, 21-19
DBMS_MACADM.UNAUTHORIZE_TTS_USER procedure, 21-37
triggers
different from object owner account, 26-19
reports, Non-Owner Object Trigger Report, 26-19
troubleshooting
access security sessions, 26-6
auditing reports, using, 26-5
factors, E-11
general diagnostic tips, E-11
locked out accounts, B-1
passwords, forgotten, B-1
realms, E-11
rule sets, E-11
rules, E-11
secure application roles, 26-6
trust levels
about, 7-10
determining for identities with GET_TRUST_LEVEL_FOR_IDENTITY, 17-25
determining with GET_TRUST_LEVEL, 17-24
factor identity, 7-10
trust levels *(continued)*
factors, 7-12
for factor and identity requested, 17-25
identities, 7-9
of current session identity, 17-24
trusted users
accounts and roles that should be limited, D-8
default for Oracle Database Vault, D-7
tutorials, 7-21
access, granting with secure application roles, 8-6
ad hoc tool access, preventing, 7-23
configuring two-person integrity (TPI), 5-16
Database Vault factors with Virtual Private Database and Oracle Label Security, 11-7
Oracle Label Security integration with Oracle Database Vault, 11-7
restricting user activities with command rules, 6-13
schema, protecting with a realm, 3-17
simulation mode, 10-8
See also examples
two-man rule security
See two-person integrity (TPI)
two-person integrity (TPI), 5-16
about, 5-16
configuring with a rule set, 5-16

U

UNAUTHORIZE_MAINTENANCE_USER procedure, 21-33
unified audit trail
how it works with Database Vault, A-1
unified auditing
in Oracle Database Vault, A-1
predefined audit policies, A-1
uninstalling Oracle Database Vault, C-2
Unwrapped PL/SQL Package Bodies Report, 26-19
upgrades
DDL operations impact, 12-3
user authorization
Database Vault authorization for ILM
granting, 21-14
revoking, 21-33
Database Vault authorization for Information Lifecycle Management
granting, 21-14
revoking, 21-33
user names
reports, Username/Password Tables Report, 26-19

USER_HISTORY$ table, *26-15*
Username/Password Tables Report, *26-19*
users
enterprise identities, finding with DVF.F$PROXY_ENTERPRISE_IDENTITY, *17-37*
enterprise-wide identities, finding with DVF.F$ENTERPRISE_IDENTITY, *17-34*
finding session user with DVF.F$SESSION_USER, *17-38*
login user name, finding with DV_LOGIN_USER, *15-15*
utility functions
See .DBMS_MACUTL package
UTL_FILE object, *26-8*
UTL_FILE package, guidelines on managing, *D-11*

V

views, *24-5*
AUDSYS.DV$CONFIGURATION_AUDIT, *24-65*
AUDSYS.DV$ENFORCEMENT_AUDIT, *24-65*
CDB_DV_STATUS, *24-5*
DBA_DV_APP_EXCEPTION, *24-6*
DBA_DV_CODE, *24-8*
DBA_DV_COMMAND_RULE, *24-9*
DBA_DV_DATAPUMP_AUTH, *24-11*
DBA_DV_DBCAPTURE_AUTH, *24-12*
DBA_DV_DBREPLAY_AUTH, *24-13*
DBA_DV_DDL_AUTH, *24-13*
DBA_DV_DICTIONARY_ACCTS, *24-14*
DBA_DV_FACTOR, *24-14*
DBA_DV_FACTOR_TYPE, *24-17*
DBA_DV_IDENTITY, *24-18*
DBA_DV_IDENTITY_MAP, *24-19*
DBA_DV_JOB_AUTH, *24-20*
DBA_DV_MAINTENANCE_AUTH, *24-21*
DBA_DV_ORADEBUG, *24-22*
DBA_DV_PATCH_ADMIN_AUDIT, *24-22*
DBA_DV_POLICY, *24-23*
DBA_DV_POLICY_LABEL, *24-24*
DBA_DV_POLICY_OBJECT, *24-25*
DBA_DV_POLICY_OWNER, *24-26*
DBA_DV_PREPROCESSOR_AUTH, *24-27*
views *(continued)*
DBA_DV_PROXY_AUTH, *24-27*
DBA_DV_PUB_PRIVS, *24-28*
DBA_DV_REALM, *24-29*
DBA_DV_REALM_AUTH, *24-31*
DBA_DV_REALM_OBJECT, *24-32*
DBA_DV_ROLE, *24-33*
DBA_DV_RULE_SET, *24-35*
DBA_DV_RULE_SET_RULE, *24-38*
DBA_DV_SIMULATION_LOG, *24-39*
DBA_DV_SQL_FIREWALL_AUTH, *24-43*
DBA_DV_STATUS, *24-42*
DBA_DV_TTS_AUTH, *24-43*
DBA_DV_USER_PRIVS, *24-44*
DBA_DV_USER_PRIVS_ALL, *24-45*
DVSYS.DBA_DV_COMMON_OPERATION_STATUS, *24-55*
DVSYS.DV$CONFIGURATION_AUDIT, *24-46*
DVSYS.DV$ENFORCEMENT_AUDIT, *24-50*
DVSYS.DV$REALM, *24-53*
DVSYS.POLICY_OWNER_COMMAND_RULE, *24-55*
DVSYS.POLICY_OWNER_POLICY, *24-57*
DVSYS.POLICY_OWNER_REALM, *24-57*
DVSYS.POLICY_OWNER_REALM_AUTH, *24-59*
DVSYS.POLICY_OWNER_REALM_OBJECT, *24-60*
DVSYS.POLICY_OWNER_RULE, *24-61*
DVSYS.POLICY_OWNER_RULE_SET, *24-62*
DVSYS.POLICY_OWNER_RULE_SET_RULE, *24-64*
See also names beginning with DVSYS.DBA_DV
VPD
See Oracle Virtual Private Database (VPD)

W

WITH ADMIN Privileges Grants Report, *26-14*
WITH ADMIN status, *26-9*
WITH GRANT clause, *26-15*
WITH GRANT Privileges Report, *26-15*

X

XStream
Database Vault role used for, *13-22*
in an Oracle Database Vault environment, *12-33*